



SURTERATEC

BASIC 3D MODELING AND ANIMATION WITH BLENDER

2006

INDEX

| | |
|---|-----------|
| INTRODUCTION..... | 4 |
| INTRODUCTION TO 3D..... | 5 |
| Modeling | 7 |
| Superficial Characteristics..... | 9 |
| Texturized | 11 |
| Lighting | 13 |
| Animation | 15 |
| Rendering | 15 |
| Formats of Exit and Compression..... | 17 |
| Postproduction | 17 |
| Overturned | 18 |
| 3D HISTORY..... | 18 |
| The 50's and 60's:..... | 18 |
| The 70's:..... | 19 |
| INTRODUCTION TO 3D EDITOR..... | 21 |
| What is Blender? | 21 |
| Blender History..... | 22 |
| Requisites, obtaining and installation..... | 24 |
| Similar conventions to the Blender official guide:..... | 25 |
| Ambience of work (What is there is?)..... | 25 |
| Windows..... | 25 |
| Sections and Panels..... | 29 |
| Toolbox..... | 29 |
| Scenarios..... | 30 |
| Navigating in 3D space..... | 31 |
| Viewer Direction..... | 31 |
| Angles of View..... | 32 |
| Displace and approach / move away the viewer:..... | 32 |
| Camera Projections:..... | 32 |
| Drawing Modes:..... | 33 |
| The 3D Pointer..... | 35 |
| Layers System..... | 35 |
| Basic Functions..... | 36 |
| Load and Save Files..... | 36 |
| User Preferences..... | 37 |
| View & Controls..... | 37 |
| Edit methods..... | 37 |
| Language & Fonts..... | 37 |

| | |
|--|----|
| <i>Themes</i> | 37 |
| <i>Autosave</i> | 37 |
| <i>System&OpenGL</i> | 38 |
| <i>File paths</i> | 38 |
| <u>Modeling</u> | 38 |
| <u>Objects</u> | 38 |
| <i>Select, scalar, rotate and move</i> | 39 |
| <i>Duplicate</i> | 40 |
| <i>Relate</i> | 40 |
| <u>Meshes</u> | 41 |
| <u>Curves</u> | 44 |
| <i>Bezier Curves</i> | 44 |
| <i>Text</i> | 46 |
| <u>Rendering</u> | 47 |
| <u>Materials and Textures</u> | 48 |
| <u>Materials</u> | 48 |
| <u>Textures</u> | 50 |
| <u>Lighting</u> | 52 |
| <u>Focus</u> | 53 |
| <u>Light</u> | 54 |
| <u>Sun</u> | 54 |
| <u>Animation</u> | 54 |
| <u>Keys</u> | 55 |
| <u>NLA Editor</u> | 56 |
| <u>IPO Curves Editor</u> | 56 |

INTRODUCTION

As time goes by and our civilization evolves, the sensorial experiences that we are in touch every day, have become a more complex art, to the point of being able to experience a world immersion that only exists in the imagination of the people. This sensorial experiences are product of high-end technologies like 3D computer graphics (3dCG), which makes possible that, at the movie theater, we could enjoy a stellar voyage or a trip to the center of the earth and even see how our deepest fears are brought alive. Another sensorial experience to which we are exposed frequently is advertising, that by means of the use of 3D, they present situations to us impossible to recreate in the real life, like for example, maracas dancing to the compass of music or the effect of a pill inside the body. Like these two examples, 3dCG have the most varied applications that could even be infinite ; but nothing of this would have been possible if it was not thanks to the success key of the human race: organization and coordination, and are these two characteristics the ones that SurTeraTec has taken as pillars to construct a network of knowledge in the 3D world and their tools, in order to present them to all those people who have the restlessness to make their ideas or dreams visible and thus to manage to transmit that knowledge.

Our personnel have gathered history, theories and useful practical knowledge of the world of modeling and 3D animation and their tools, Blender in this case, in order to present them in an understandable way, (making special effort to transmit the information from a simplistic or minimalistic point of view) with the purpose of creating the theoretical and practical knowledge base necessary to advance further in the 3DCG.

INTRODUCTION TO 3D WORLDS

When we look for the key factors to see a 3D world in a computer, we consider a series of characteristics. These influence in a decisive way our perception of the 3D artificial world in which simply we worked or played. Let us see which are those:

- When we watch a computer screen what we see is a 2D surface, we needed to generate the sensation of third dimension. All the objects that we show will have to keep an order in depth towards the screen, or what is the same, to keep a certain Z coordinate. Other effects like shades helps to create deepness in the scene.
- The scene that we see must be interactive, in that manner, it must respond to our actions; this is obtained by drawing the scene in real time. Whenever we move in the scene, this one is drawn different according to the angle of view.
- **Belief:** It could be possible that what it is shown in the screen it was not realistic or it does not have all the characteristics of a real object. Simplifications can be done so that the scene seems real although it is not realistic, as it is observed in figure 1.

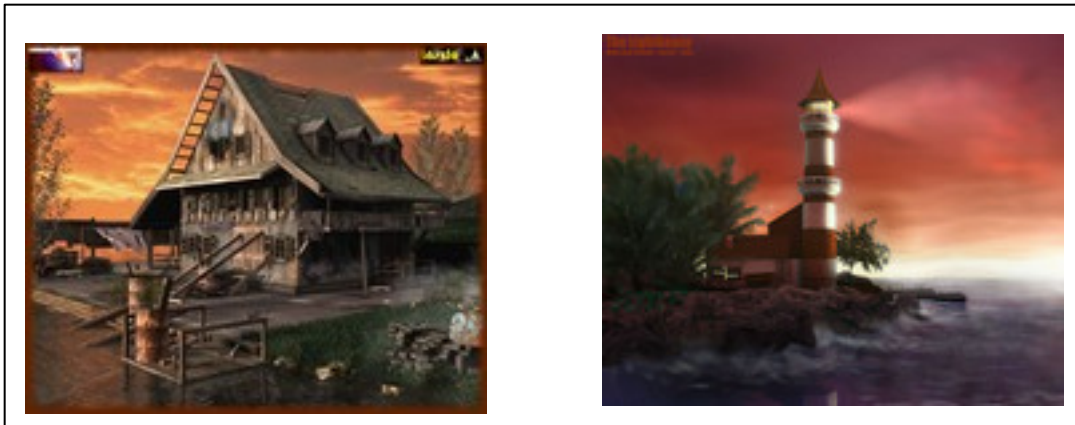


Figure 1. Simplified scenes.

Normally when we watch an object in the real world this one has a perfect definition, the real objects have infinite details, as we look closer to it, more details will arise. On the other hand, the computer must save the artificial world in a digital form; furthermore it should allow to move it, light it and render it with textures without pauses so that it gives a sensation of fluid movement, and this is the reason why the definition of the details and finally the realism is limited by the hardware.

Another point to take into account, is that the computer has a limited memory and it can not store all information that a real scene contains, therefore in the process of digitalization of a real scene, information is lost. When we generate an artificial scene, it is always trying that it has the maximum of real world information in order to increase the

level of detail. The objective is to balance the level of details with the facility of manipulation of the scene, so that we can move by it in real time without a predefined patterns.

The problem is to keep the maximum of that information, with the minimum of data to store and process. In order to reach this objective, the world is divided in sets of objects and properties are assigned to each object. At the same time, each object is made of basic units called primitives, for example a triangle. The primitives are like the bricks to construct the scene. There are several types:

- **3D pixels:** also called voxels
- **Vectors:** They indicate a direction
- **Polygons:** triangles, squares
- **Primitive of volume:** spheres, cones, cylinders

One of the first things that it would be necessary to clarify is the difference between a 3D animation and a traditional animation (like a classic *Disney* movie):

- In a **CARTOON** we have: **drawings** that pass by in front of our eyes at high speed, concretely with a rate of 24 images every second (**24 fps** / frames per second). Actually any movie is exactly the same, but in that case would be 24 photos taken from a real world scene.

Simplifying the process: a traditional animator has to draw each frame one by one. Draw, ink and then apply colors ... Normally it is worked on a transparent material, that way, a more elaborated static background could be seen.

- In a 3D animation, however, we are not drawing. What we do is to build, **to shape in 3 dimensions** each of the elements, actors or scenarios that appear in all the scenes. The computer and all the different tools (software) that we use, allow us to generate these shapes, to apply all kinds of superficial characteristics, light the scene and move everything there is, whether it is an actor, a light source or a camera.

The big difference is that, in 3d it is not necessary to create each position of the every object for every frame, because once it is created, we can see it from any point of view. Although we are talking about stages and virtual actors, they have a three-dimensional nature.

We have explained superficially a process that indeed is really complex. We are going to see it with more detail further on.

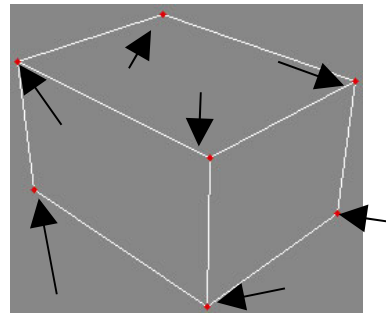
Modeling

Once the plot is set, after planning every scene, draw multitude of sketches and other previous tests, is that begins the process of the modeling: the creation of the three-dimensional structure of every element.

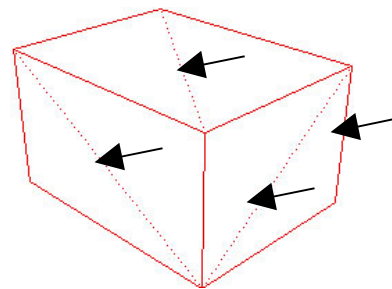
If we look around and analyze the form of the objects, we will see that an enormous variety exists. The work of a 3d modeler begins analyzing each of the basic shapes that defines an object. A ball of soccer is a sphere, a canister of meal can be a cylinder and a dice is a cube. These are simple objects based on basic forms (generally called “primitives”).

Defining what sides, edges, vertexes and polygons are:

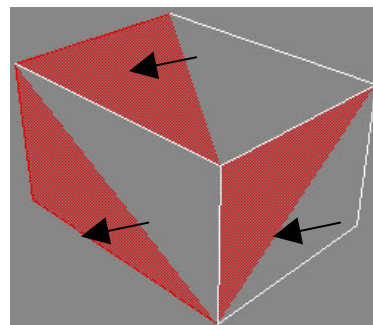
Vertex: Point in the space, on which two or more edges will converge and define the structure of the objects.



Edge: It is the straight line that represents the rim or the intersection of two planes. Also it is possible to see it as the union of two vertexes.



Face: Plane containing 3 edges and 3 vertexes. It is the fundamental unit in the construction of objects in 3d, because it is the minimal representation of a plane in the space.



In our course, we will be calling object to every element in the virtual space that has at least 2 vertex and an edge. Example of objects we have spheres, cubes, cones, planes or nurbs meshes, etc. as in figure 2.

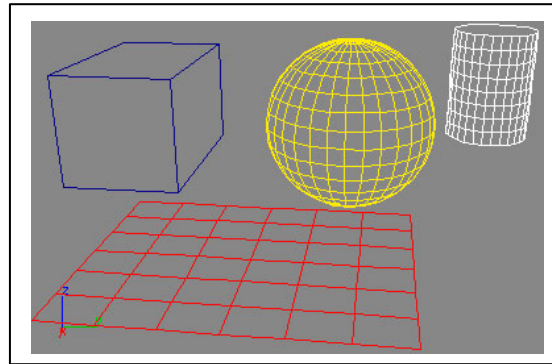


Figure 2. Example of 3D objects.

But a big part of the objects consist of several basic forms. We can see a funnel as the intersection between a cone and a narrow cylinder; a bookstore or a building consist of many parallelograms (blocks) of different thickness and heights, and finally there exists much more difficult objects to shape, from objects with multitude of curves (as a sport car) up to the organic forms of all the living beings (a tree, a rose, a cat or a person) and in general, of the nature that surrounds us (mountains and geologic formations, water in movement, the clouds). For all this situations, the developers of software have had to create generally quite complex systems of modeling (nurbs curves and control meshes, systems of particles, dynamic simulations, etc.)

Initially (and still today in many programs of modeling) the system used by the computer to represent every structure is the **polygon**. A cube has 6 sides, each side of them is a polygon—a square—; a pyramid like the ones we find in Egypt, consists of 4 triangles and a square base. However a round shape is also represented by means of polygons; the clearest example of the real life can be seen in a soccer ball, that it is built out of 12 pentagons and 20 hexagons.

Normally, the curved surfaces are represented by means of triangles. Some programs can work with polygons of any number of sides, but others do not. The big "advantage" of a triangle is that its 3 vertexes are always in the same plane (this is the reason why a 3 legged chair can never limp) however, in a square polygon or higher, the vertexes can move accidentally giving place to a non planar shape, which can cause internal conflicts that affect to the good visualization of the surface.

Nowadays other systems of modeling exist, where the user does not work with polygons, but with **curved surfaces defined mathematically**. Let's imagine a

circumference: it might be represented as a polygon of many sides, but also it might be represented as a mathematical function between two variables X and Y (group of points of a plane equidistant of other). Clearly the user does not have to use bothersome formulas, but likewise in a vector graphics software it turns out to be simple to draw perfect curves (not only circles or ellipses), in a non polygonal modeler there are different types of tools (splines, NURBS, patches Bezier, etc.) to create complex superficial curves.

When we work with polygons it is necessary to **economize**, it is not worth the use of a lot of polygons to define a curved surface (a simple sphere) if it is going to be observed from far away. The creators of games optimize the number of polygons of their objects to be able to process this information in real time, it is always necessary to be aware that the computer must take an absolute control where is every vertex of a polygon located at every moment, and the number of these points in the space can go up with relative facility to several millions.

The advantage of splines (mathematics curves) is that they always define the surface perfectly. However, they turn out to be more difficult to handle and in certain situations it can turn out to be very bothersome to solve some problems with them.

Superficial Characteristics

If we look again around us we will verify that together to the structure of the objects, there is an enormous **variety of superficial characteristics**, and all of this must be simulated in the computer.

As soon as the model is finished we must go to each of its parts or pieces and assign them different properties like:

- **Color:** this is perhaps what we perceive more clearly. And, however, not always it is something so simple: what color is a mirror? And a glass? And our skin? Normally more than one variable manages to define the color, as the **diffusion**, that controls the quantity and the color of the light dispersed by the object, or the **ambiance color** that controls the sensibility of the material to the **ambiance's light** (basically we control the quantity of light that there is present in the shades of an object, since hardly ever they turn out to be black).
- **Specularity:** controls the sheens or twinkles that the light produces in an object. An object is very brilliant if it has a high specularity and matt if it has low.
- **Reflectivity:** controls the reflexes of the environment in the surface of the object. Often when we look at an object we do not see the color of this material, but what it reflects (the most extreme case would be a mirror). The surface of a new car is reflecting, that of a jean cloth does not. Normally a high reflecting object also is very brilliant (specular).

- **Transparency:** a glass of our window will allow us to see what is on the other side (if it is clean). If there were not any other factors we would not have to see the crystal, what happens is that sometimes it is dyed and almost always we distinguish the crystal for the reflexes that emits, the twinkles of light or the distortions that take place on having looked thru.
- **Refraction:** these distortions in the path of the light that we see are the result of a process of refraction. The crystal of a magnifying glass deforms whatever there is below (increasing it) by a refraction process. A stick into the water seems to double, for the same motive.

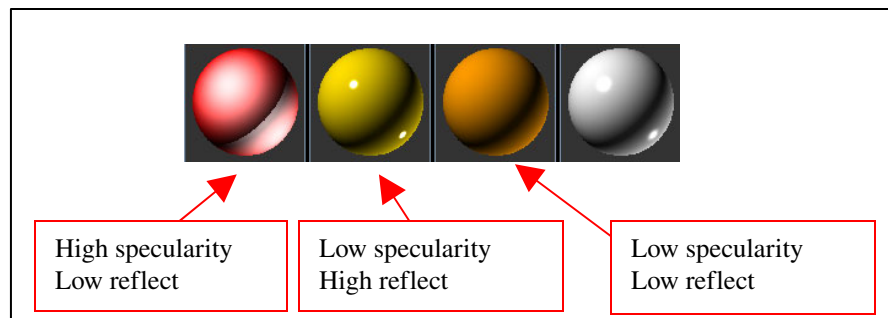


Figure 3. Superficial Characteristics.

There are other properties (luminance, causticity, anisotropic characteristics, etc.) but the previous ones are the most important. The different 3D applications allow us to control these parameters and the realism of a scene depends on their tuning in the material applied. It doesn't matter how perfect the object has been modeled, it one can lose all its credibility if the color is too saturated or if all the surfaces are too brilliant and reflecting (very common defects in the works of many begginers).

Textures

This should be a subsection of the previous point, but because of its importance we will study it apart.

Many objects cannot be defined only by the superficial color. The top of the ground, the wood of the furniture or the pattern of a shirt, consist of different colors with a, sometimes geometric distribution and others completely random, that is why we resort to the textures.

If we digitize a marble piece and keep the image with a certain format, later we can apply that superficial finish to every object. And it doesn't have to be strictly flat: we can apply it to a cylinder, to a sphere or anything we want, applying it in a way that the image covers completely the whole surface or that repeats progressively.

This type of texture (generally a real image or one created by us in an image manipulation program) it is known as a bitmap texture. As in every bitmap image (like a picture) is very important to control the **resolution**, adapting it to our needs; if we do not do so, it would happen that on zoom in the pixels of the image would appear.

There are basically 4 methods to apply a texture:

- **Planar:** To apply a marble texture on a virtual floor, for example, if we apply this method in an object we will see that in the face where it is applied it is draw perfectly, but in the adjacent ones it turns out to be projected longitudinally.
- **Cubic:** To avoid the previous problem, we can use this system. If we have to texture a closet we would do it by means of a cubic application, the texture then is being projected in all six directions of the faces of a cube.
- **Cylindrical:** If we want to put the tag to a bottle of wine we will use a cylindrical projection.
- **Spherical:** To apply the texture of the seas and continents to an esphere in order to recreate the earth, this would be the suitable procedure.

Clearly there are many objects that are not quite these forms, and where we do not see any of these methods of texturizing so clear (i.e. an animal?). And there is where talent takes control: sometimes we can split an object in more basic different areas, other times we can texture a piece before a distortion is applied ... All the aspects of an image influence the quality of it, but perhaps is the texture what it is the most important. A good texture can save a mediocre modeling (in fact the video games base his quality more on texturing than on modeling).

Other methods are called **procedural** or **shaders**. It is a matter of a few internal algorithms that the 3D software process, normally departing from fractals structures (mathematics structures), that contribute different benefits:

- The resolution is always ideal (we never get to see pixels).
- Because of its nature, fractals normally imitate very well the chaotic details of the nature (as the crust of a tree, the seams of a marble or the flames of the fire).
- We never perceive phenomena of repetition (something very disagreeable but regrettably very much in use, making that a small bitmap texture repeats in all directions and demonstrating how artificial the image is).
- Normally the calculations made by a computer are faster than when a very big map of bits is applied (anyhow some shaders can become very complex and, therefore, not so fast).

When we use shaders, many of them are applied in all directions and covers perfectly all surface (other advantage of this type of textures).

In any other case, there exists more complex methods of texturing, like the **UV**, that takes into account how the object has been generated in the phase of modeling (following generation coordinates) in order to apply the texture, adapting itself to the shape like a glove.

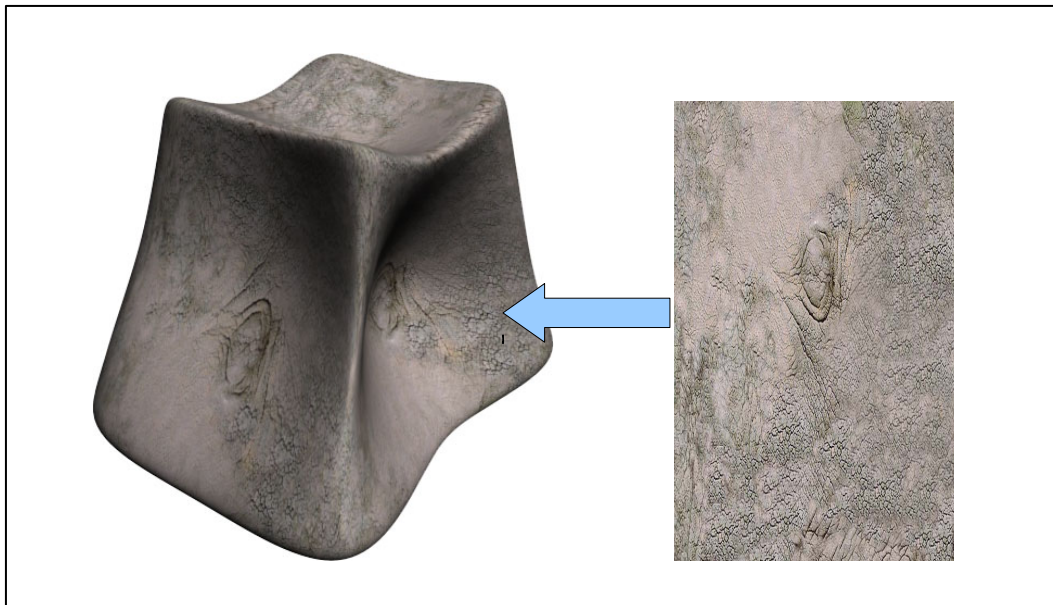


Figure 4. Rhino texture applied to an abstract object.

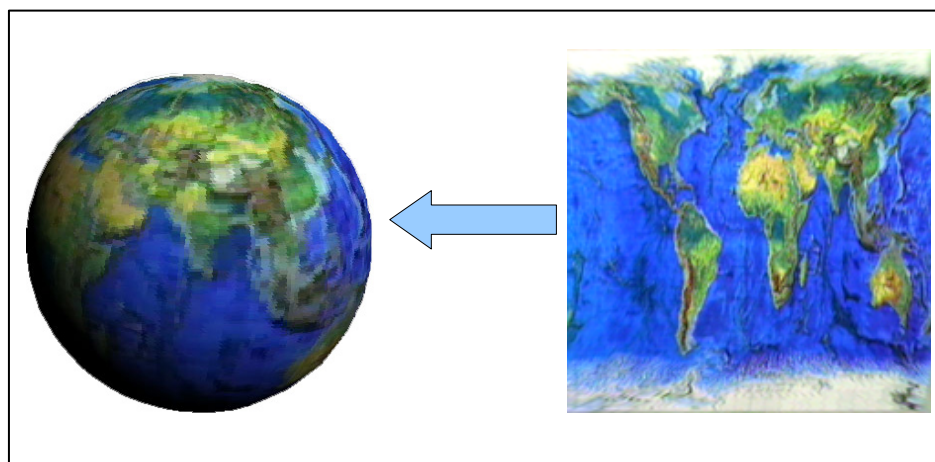


Figure 5. Texture of terrestrial surface applied to a sphere.

Lighting

This is one of the most difficult disciplines of the computer graphics, since in the real world the light has a complex behavior that does not turn out to be easy to imitate in our computer. The principal difficulty derives from the fact that the light is emitted from a certain point (the Sun, a bulb, the flame of a candle...) and on having collided with the bodies, it illuminates them, but also it reflects on them, lighting other points that, at first, it would seem that they should not turn affected by it.

In any 3D software we have different types of lights to illuminate a scene. In general one always speaks about 4 classes of lights (there are others, but these are the most important):

- **Radial:** A light that comes from a certain point — that we place in the scene — and it emits its beams in all directions. It would be the suitable light for a bulb that hangs by the wall, or a flame ...
- **Spot:** The typical lights of the theaters or spectacles. They are directed in a defined direction and we can control the opening of the cone of light, as well as its diffusion (if it is sharp or smooth) as well as other factors.
- **Parallel:** Is the suitable light to simulate our Sun. This is a star that is placed in a specific point and emits light in all directions, and for that we might use a radial light to represent it. But respect to us, the Sun is so, so far away, so much, that placing a luminous point so far away in our virtual world does not turn out to be useful. That's why we have this type of lights: they are called parallel because although we place them at very little distance of our scene the beams that they emit are parallel, like —practically— are those of the Sun when they reach out the Earth.
- **Ambience:** Is a type of light that does not come from any specific point. It comes from all directions. As we have said, the light not only comes from a certain point and it reaches an object in a direction, it lights it from a certain angle, and also it bounces. In a room with white walls (or light colors) the light that comes in from a window (it means: from certain direction) bounces in all walls and objects that are in the path of the beam of light, so that we can meet a couch that is lightly illuminated in an area in the one that should be in shade (notice that in a room, hardly ever we will see areas to 100 % of darkness —black—). Outdoors also another phenomenon happens, that is the dispersion of the light on having crossed the atmosphere, the clouds or the pollution. To be able to simulate this type of effects the lights called *ambience*, were created. Anyhow we will see that there exist certain types of systems that takes into account the phenomena of reflection of light (radiosity), although they turn out to be enormously slow, for the big quantity of calculations that the computer must process.

Of course in any type of light it can be controlled a great number of parameters, like: intensity, color, attenuation with the distance (“dropoff”), beams, halo, and lens flares, etc.

Directly related to the lights are the shades generated by the objects. In the real world every light that we analyze projects a shade when it comes across an obstacle, although it is a question of reflected lights. In a 3D program, on the other hand, it is possible to control a light so it does not project shades, in order to save calculations. We must think that the computer graphics is a **constant trick to our eyes** and any trick is valid as long as we save time of calculation and the quality of the image does not suffer substantially.

Animation

If we notice, we are exhibiting a process similar to that of the movies or the photo: we construct stages, give them a hand of painting, illuminate them appropriately and all there's left is hire good actors and photograph or roll the action.

In certain animations the movement remains limited only to a series of "flights" around or inside a stage. The most obvious example is the architectural computer graphics. For a 3D program a camera is just another object which it can be added into our stage, giving it certain characteristics (image format, opening of focus ...) to capture the virtual environment from its virtual lens. Normally we can add so many cameras as wanted and move them to our will, with the advantage that we do not have limitations like the heavy real world cameras.

Things starts to complicate if we want to move other objects. Some times it is so simple as to displace or to turn an element: most of the times we have mobile pieces that they move and rotate a little with others as reference and in the most complex cases we are talking of moving characters, making them walk, run, laugh, shout or cry ... Then we are already speaking about an enormously complex discipline. The ants of “*A Bug's Life*” or the toys of “*Toy Story*”, the dinosaurs of “*Jurassic Park*” and the characters of “*The Phantom Menace*”, they represent the edge in this field. Perfection obtained with the effort of many persons, very specialized, working during a lot of time, with the best machines and programs.

For a basic animation only it is necessary to have an object that we want it to have some movement, after all, why making a video if nothing moves? i.e. We can make a minimal animation, with a box that it turns on its own axis or a sphere with a texture of the surface of the earth which it is rotated giving the impression of the rotational movement of the planet earth.

In the world of the 3D animation, there are some characteristics that have survived from the traditional animation like it is to work with pictures that are being changed at an appropriate speed and gives us the sensation of movement. Here we do not draw picture to picture, instead of that, we concentrate in specifying the significant changes of the

movement, that is, if we have an animation with 100 pictures where a box moves from one place to another, we would create the box in the picture 1 and in the picture 100 we place it in the final position; the 3D software, will be in charge of "drawing" all the intermediate pictures.

Rendering

Up to this moment it is us the ones that have worked and now it is turn for the machine. As we were saying at the beginning: in 1 second of video we have 24 stills (in the PAL system there are 25 fps and in the NTSC 30 fps) and the computer must calculate each of these images. This process is called Rendering.

There exist several methods (algorithms) for rendering, but the most important are:

- **Wireframe:** normally it is used to do tests of movement, to see how the things are advancing and not to encounter any surprise at least in the movement of the objects. It is the fastest method, and what it shows is only a few lines that define the polygons of every element. We do not distinguish any type of texture but only the tructure of the objects (as when we are shaping), but it has a great usefulness to test the quality of the movements in an animation before using other much slower methods. Normally we see both the front (visible) structure and rear (invisible) of the objects. There is a variant called "secret lines", which allows concealing the back of the objects or the elements that happen behind others.
- **Phong:** in several programs this is a quite coarse algorithm, which cannot even represent shades or other physical phenomena and it is also used only to prove the animation. But due to its high speed some programs have turned it into its higher level rendering engine, implementing some services to balance these lacks. In fact, and in spite of its many limitations, it is most used in big productions, where the time of rendering cannot go off excessively.
- **Raytracing:** here the reflections, the projected shades or the refractions are calculated in accordance with similar parameters of the real world giving a result that comes closer to reality. The counterpart is that it turns out to be much slower than Phong and normally it is used more in static images or stills than in animations. In this method every visual beam that goes out of the camera arrives to the objects and, according to the indexes of reflection, transparency or refraction, it goes then from here to other objects or lights in the scene. Every visual beam that goes out of our camera will correspond to a pixel (minimal unit of visual information) of our image.

- **Radiosity:** it is the most perfect of all the methods of rendering, but it is also the slowest. Here also are calculated the interactions between the light and the color of objects nearby, if (i.e) we place a red ball near to a white wall we will see an area of the wall nearest to the ball as it is dyed of red. Another example: if we illuminate a wall, this one reflects part of this light providing a more tenuous light towards the objects that are closer to it. This is a perfect system for very realistic simulations in the field of the architecture, especially in interiors design, since it best simulates the behavior of the light in these conditions. Also it is used very often to create the stages of some 3D video games to contribute realism (with the peculiarity that the scene is previously calculated and saved on the disc, otherwise it would be impossible to play in real-time...)

Once rendering a sequence we can record it directly to video or what happens more often, input this file to a program of post-production to retouch some details before saving it as video (or film).

Video formats and Compression

After having rendered the total of the images that belongs to the animated sequence, it is time to move forward to the process of construction of the video as such. This is nothing more than to "pile up" the images one after another in a single video file which extension is generally: .AVI, .MOV and .MPEG or .MPG, so that the multimedia player will be able to show the animated sequence. To calculate how does it costs to create and to store a file of video is something simple, first it is necessary to know what size the picture is going to be in pixels, 100x100, 300x400, 640x480, 800x600 etc., let's suppose that it is 640x480 pixels, by multiplying these two values it would give us a plane of 480.000 pixels, then we define all the possible colors it might have every single pixel or what is the same the depth of color, we choose a 256 colors palette, and so, 1 byte is needed in memory to store the information of the color for a single pixel. That makes 480KB per picture or frame and assuming that we are using a speed of 30 frames per second, 300 frames are needed to complete 10 seconds of animation. Joining all those numbers, we would have that 144MB are needed to store the video segment, this is something exorbitant if it is taken into account that is a video of only 10 seconds with not very high quantity of colors and a low resolution of screen. It is for this reason that a compression factor is used in every frame to be able reduce its size. There exists different algorithms of compression called video codecs and some of these are:

- Cinepac of Radius
- Indeo Video of Intel
- MPEG-4

- Video of Microsoft
- Xvid
- DivX

Using some of these codecs might go as far as to compress every frame of our previous example to 24KB or what is the same, using a compression ratio of 20 to 1, having a total of 7MB for the same ten seconds of animation, which is still somewhat a high value, but it is more efficient than without compression.

Post-production

Sometimes it is interesting to adjust some characteristics of the image or animation once rendered, do simple things like saturating the color or raising brightness, make focus effects ... or more complex ones like mixing different parts of an animation where the foreground could be worked apart from the background, or to integrate any image in a real scene. Whether if we are talking about a static image or an animation, there exists numerous programs that helps to make those changes, *Photoshop* and *AfterEffects* are just two commonly known.

3D HISTORY

The 50's and 60's:

A combined effort from the companies International Business Machines (IBM) and General Motors (GM), resulted in the creation of a program called Design Augmented by Computers (DAC-1). The main goal of DAC-1 was to accept coordinates and draw the representation of those coordinates. Once the data was loaded it was possible for the designers and engineers to see the designed model from different directions and angles. Although the program DAC-1 was created in 1959 it was not disclosed until the 1964 Detroit conference.

In 1962 Ivan Sutherland of the Massachusetts Institute of Technology (MIT) wrote a program that controlled directly a cathode rays tube (CRT) to emit lines and light, that displayed a geometric figure as the result. The program called "sketchpad" allowed to save the drawings that were created. The drawings created with Sutherland's program were vectors instead of bitmaps, the results were quite precise since the user was specifying the geometric figure, and certain indications (if it was a rectangle, for example, only had to define two corners), the computer calculated automatically the size and position of the lines. Sketchpad propitiated the birth of the graphics science controlled by computer.

Two years later, Sutherland would collaborate with the doctor David Evans to initiate the exploration of mixing art and science (computational). It was the University of Utah the first one that had an academic laboratory specifically to develop computer graphics. Still today many of the graphic softwares take its bases from the investigation that took place in the University of Utah, from those of 2D design up to those of virtual reality. It did not take a lot of time before the companies were beginning to be interested in computer graphics, IBM, for example, launched to market the IBM 2250, the first commercial computer with a graphic system. The Magnavox Company, as well obtained the license to distribute a video game system created by Ralph Baer, the product was named Odyssey. The Odyssey was the first product targeted to the consumer market with computer generated graphics.

Dave Evans was hired by the University of Utah to create the laboratory of computer science. Evans's main interest was to develop computer graphics. Evans hired Sutherland, and is in Utah where Sutherland finally reaches a high degree of perfection on a HMD (head mounted display) interface that he had developed some years earlier. In this period, Evans and Sutherland were frequently advisers of companies; nevertheless, constantly they felt frustrated by the absence of technology and that was the reason for founding their own company.

The 70's:

A student Sutherland's class in the University of Utah, Edwin Catmull, glimpsed the computer animation as a natural evolution of the traditional animation and while still in Utah he created an animation, it was his hand opening and closing. One of his targets turned in making a movie completely generated by computer. From the University of Utah a big technological advance arose in the field, John Warnock was one of the digital pioneers, and he founded one of the most important companies that changed the course of the history as long as it is in the field of digital design, he founded Adobe. Another graduated from the University of Utah is not less known, Jim Clark, founder of Silicon Graphics Inc. (SGI).

1970 also marked a revolution in the television market. Companies as the CBS started using products developed to make computer animations. The Computer Image Corporation company (CIC) developed combinations of Hardware and Software to accelerate processes of traditional animation, by digital means. CIC was offering ANIMAC, SCANIMATE and CAESAR, with these programs the drawings could be scanned, trajectories created and apply principles of traditional animation as stretching and shrinking.

In the field of the 3D animation, there was created a new type of digital representation, the algorithm of Henri Gouraud. This one allows the outlines of the polygons not to be pixelated, since that was destroying the sensation of a soft surface. The algorithm creates the interpolation of color among polygons and thus it achieves a better representation of curved surfaces. The advantage over the traditional method (the flat render) is that the surface in effect seems to lose hardness, with only a small penalization in the time that takes to render.

In 1971 the microprocessor arises, using integrated circuits technology, the electronic components were miniaturized. The Atari Company was created and in 1972 it creates the first arcade video game machine, Pong. Evans and Sutherland (E&S) were making already hardware of their own to avoid some of the technological limitations that they had experimented before. One of the most impressive systems created precisely for E&S was "Picture System ", it included a graphic tablet and a color buffer. In 1974, Triple-I developed a system to film the images generated in a computer. Other of its inventions was the creation of graphics acceleration cards. The developments of Triple-I were a big advance that was allowing the synthetic graphs to be used in movies.

After his graduation, Catmull was hired by the company Applicon, where he did not last too long, since he was proposed to found the laboratory of computer animation at the New York Institute of Technology (NYIT). Some of the workers of the University of Utah were also invited and they accepted the work in the NYIT. The first animation programs developed inside the NYIT were to support the traditional animation. The first application that Catmull developed was "tween", that allowed to do the interpolation between pictures. Also there was developed a scanning and painting system that later turned into Disney's production system, the CAPS (Computer Animation Production System). The NYIT created a department dedicated to the investigation of 3D graphics, and for two years its main project was to create a movie, " the works ", that was never concluded, and in

fact, the preliminary tests were quite discouraging. Before the defeat of short "Tubby the tuba ", several employees went out of the NYIT. Apparently the director of the institute never accepted that they were hiring film directors to create the movie, reason for which the result was not better than it was possible to have obtained. James Blinn, developed a similar algorithm of texturized, but instead of representing color it was representing depth. The mapping colors provoke that the surface has a relief or a depression. The white parts of the image are represented as nodes, while the dark parts represent the depressions. Providing of textures and reliefs can create quite realistic models. The algorithm was named " bump map ". Another algorithm presented by Blinn is that of reflectivity, with which a reflex of the ambience is simulated in the one that finds the object.

Of the University of Cornell, Rob Cook raised a new algorithm that it was eradicating some of the limitations of the previous representations. Cook appreciated that the representations of the epoch were of plastic appearance. Using the variable of luminous energy that emits the virtual light he could create a material that looks like to a polished metal. The previous methods were considering the sheen of the synthetic light.

Ed Catmull based his PHD's final project on an ingenious and brand new way to represent surfaces. This new technique called z-buffer helps in the process of hiding the parts of the scene that will not be seen by the user in the final representation. In addition to the z-buffer, Catmull included a new concept, texture mapping. History says that talking with one of his partners, Catmull thought that if an object in real world could be painted to make some other representation; in a virtual world there was no reason not to do it.

The French mathematician Dr. Benoit Mandelbrot published an essay that allowed adding realism to the scenes generated by computer. The document "A Theory of Fractal Sets", explains that a line is a unidimensional object, the plane is a two dimensional space; nevertheless, if the line describes a curve so that it covers the surface of the plane it stops being unidimensional, although it is not two dimensional either. Dr. Mandelbrot referred to this space as a fractional dimension. The main applications that were given to the theories of Mandelbrot were the creation of random areas, as well as the creation of textures in which subdivisions exists inside the same pattern.

INTRODUCTION TO THE 3D EDITOR

What is Blender?

Blender is a program that allows the user to create, modify and animate objects in a 3D world in order to create video with a 3D content, as well as its post-production and playback.

Some of the advantages of using the Blender as a working tool are:

- **Distribution:** With only 7.04MB (version 2.42a) it has all the functionality of professional commercial products, which sizes are over 100MB; it is clear from here the high efficiency in the distribution over the Internet.
- **Portability:** Blender's user interface (UI) has been designed with OpenGL and so, it guarantees the the same working environment whether the operating system is Windows, MacOS or Linux.
- **Speed:** Due to its render speeds, and the high objects count (included textures and lighting), Blender has gained the respect of the 3D developers' community and created a culture which avoided the disappearance of the product after the management transition from a company to a foundation.
- **Cost:** Blender is free of charge, because it is protected by the general public license (GPL), which stipulates among other things that any program that is under this type of license must be of free distribution.
- **Honesty:** Even though it is not common to describe a software as honest, Blender has this quality because it is a GPL software with open source, making sure that there is no "malicious" code, like for example some that allows access to the user's private data.
- **Versatility:** The UI is completely configurable since it is based on a type of windows that can be joined or divided in accordance with the needs of the user and once it is set, the "screen system" could be used in order to have multiple configurations available immediately with the use of the keyboard, offering this way more than one UI adapted to every task there is in the process of creation of an animation or 3D world.
- **Support:** Blender's big community, which you will be soon part of it, offers a enormous knowledge base freely available to help accomplish those tasks planned but still does not know how to do it. This community is being conformed by news servers, mailing lists, enthusiast's web pages at diverse levels of experience (from the beginners up to the gurus of 3D), FTP servers etc

Blender History

In 1988 Ton Roosendaal founded along with other associates the animation studio NeoGeo and rapidly turned into the biggest in Holland and one of the principal animation house in Europe. While at NeoGeo Ton was responsible for the artistic direction and software development. After debating it meticulously, he arrived to the conclusion that the set of tools at NeoGeo were obsolete and that its update was completely inviable, so they came to the conclusion of coding everything from scratch. In 1995 this rewriting began, which destiny was to turn into the suite that at present is known as Blender. As NeoGeo kept on refining and improving Blender it turned out obvious for Ton that this one might be used by other artists out of the company.

In 1998, Ton decided to found its own company called "Not to Number" (NaN) to be able to develop and trade Blender. The principal idea of NaN was to create and distribute a free, compact and multi platform tool. At that moment that was a revolutionary concept because most of the commercial 3D software were costing hundreds of USD. The business model that NaN implemented was about providing products and commercial services concerning Blender. In 1999 NaN participated at its first conference in Siggraph in an effort to promote a major use of the program. This was a success and it attracted a big attention as much of the press as of those who were present at the conference, it was a fact, Blender was a success and it was confirming a big potential.

After Siggraph in early 2000, NaN assured the financing for 4.5 million euros by means of venture capital. This amount of money allowed the company to grow rapidly and to expand its operations. Soon NaN grew with 50 employees all around the world, trying to improve and to promote the program. In the summer of 2000, a second version of Blender was released to the public. This version added the integration of a 3D game engine. At the end of 2000, the number of registered users was already exceeding 250.000 users.

Unfortunately, the ambitions and opportunities of the company, were not corresponding neither to the capacities of thee nor to the realities of the market at that moment, but a rapid growth of the community, helped a restart of NaN with new investors and a resize (smaller) of the company in April, 2001. Six months later the first commercial product of NaN called Blender Publisher was released. This one was oriented to the emergent market of the 3D interactive content for the web, but due to the disappointing sales and the difficult economic climate of the company, the investors decided shut all the operations down. This closing also involved stopping any further development of Blender. Nevertheless, Ton could not left abandoned Blender although this one had some disadvantages, as a complex internal architecture, not finished characteristics and a unusual user interface, and since opening a new company with a team of developers with a sufficient number of members was not possible, in March, 2002 Ton Roosendaal founded the non profit organization named *Blender Foundation*.

The main goal of this foundation was to find a way to continue the development and promotion of the Blender as a project in the open source community. In July of 2002, Ton came to an agreement with NaN's investors and as a result there was a campaign to collect 100.000 euros called "Free Blender" with which they would pay the license for the source code and the copyright to NaN's investors. With a group of enthusiasts, among them several

NAN's ex-employees, it just took only 7 weeks to reach the goal and on Sunday October 13th 2002 the first version of Blender was released to the public under the GPL and from there to the present it is being maintained and developed by a team of dedicated volunteers all around the world leaded by the original creator of Blender.

Blender's development log:

| Version | Date | Event |
|---------|----------------------|--|
| 1.00 | January 1995 | Blender development in NeoGeo studio |
| 1.23 | January 1998 | SGI versión published in the web. IrisGL. |
| 1.30 | April 1998 | Linux version and FreeBSD in OpenGL and X |
| 1.3x | June 1998 | NaN's foundation |
| 1.4x | September 1998 | Alpha version for Linux and Sun |
| 1.50 | November 1998 | The first Blender's manual is published |
| 1.60 | April 1999 | Windows version with pay characteristics (C-Key) |
| 1.6x | June 1999 | BeOS and PPC versions. |
| 1.80 | June 2000 | End of use of C-Key. Free again. |
| 2.00 | August 2000 | Real time and interactive engine of 3D |
| 2.10 | Dicember 2000 | New engine, physics and Pyton |
| 2.20 | August 2001 | System of animation of characters (characters) |
| 2.21 | October 2001 | Launched <i>Blender Publisher</i> |
| 2.2x | Dicember del 2001 | Mac OSX version |
| | October 13th of 2002 | Blender pass to Open Code. First Conference |
| 2.25 | October 2002 | <i>Blender Publisher</i> happens to be free |
| | October 2002 | An experimental branch of Blender is created for programmers |
| 2.26 | February 2003 | First Blender totally Open Code |
| 2.27 | May 2003 | Second Blender totally Open Code |
| 2.28x | July 2003 | First Blender of the serie 2.28x |
| 2.30 | October 2003 | The interface of 2.3x version was presented. Second conference. |
| 2.31 | Dicember 2003 | Update of stable interface of the project 2.3x |
| 2.32 | January 2004 | Optimized the internal capacities of rendering |
| 2.33 | April 2004 | Ambient Occlusion, game engine is back |
| 2.34 | August 2004 | Particles and force deflection, new ramp shading control |
| 2.35 | November 2004 | Undo, object hooks, particle duplicators |
| 2.36 | February 2005 | Improvement of normals and textures |
| 2.37 | June 2005 | Inclusion of Soft Body system |
| 2.37a | June 2005 | 2.37 version revised. |
| 2.40 | December 2005 | New Character animation tools, Modifiers, Particles, Fluids, Booleans, ... |
| 2.41 | January 2006 | Better editing and playback of interactive 3d playback |
| 2.42 | July 2006 | Array modifier, vector blur, new physics engine |
| 2.42a | July 2006 | 2.42 debugged |

Requirement, download and install

As long as the 2.42a version is being used, these are the requirements:

1. 300MHz CPU.
2. 128MB RAM
3. OpenGL graphics card with 8MB of RAM
4. Hard disc: 20MB

To download blender, go to the following URL:

www.Blender.org

This material uses Microsoft Windows as the chosen operative system, so you will need to download the appropriate version of Blender. Then proceed as any other installation, open the installer doing double click on it and follow the instructions of the installation wizard. The process is really fast where there is only a step to configure: adding or not an icon in the desktop, an icon in the start menu and the association of the blende files with the Blender. The rest is standard, as in any other installation: select the path for the installation and start the program after having finished the installation.

Similar conventions to the Blender official guide:

- **RMB:** Right Mouse Button.
- **LMB:** Left Mouse Button.
- **MMB:** Middle Mouse Button.
- **SB:** Space Bar.
- **UA:** Up arrow.
- **DA:** Down arrow.
- **LA:** Left arrow.
- **RA:** Right arrow.

Working environment (What is out there?)

Blender has an user interface that at first glance feigns to be something very confusing because of the quantity and size of the buttons, added to this is the fact of not using standard controls (buttons, text boxes , radio buttons) from Windows or Mac, and that makes it more difficult to understand. Nevertheless and even though that the learning curve is high, once it is controlled it turns out tremendously practical.

There are some notorious not conventional basics for the interface. These are:

- **RMB** to select, except the file and image browser.
- **LMB** to place the 3D cursor or the animation frame.
- **SB** to show the toolbox.

Windows

Every window has a toolbar that in blender it is known as "Header", which can be placed in the top or bottom of the window and even could be possible to work without it. This one changes depending to the type of window that has been selected. The rest is the working area.

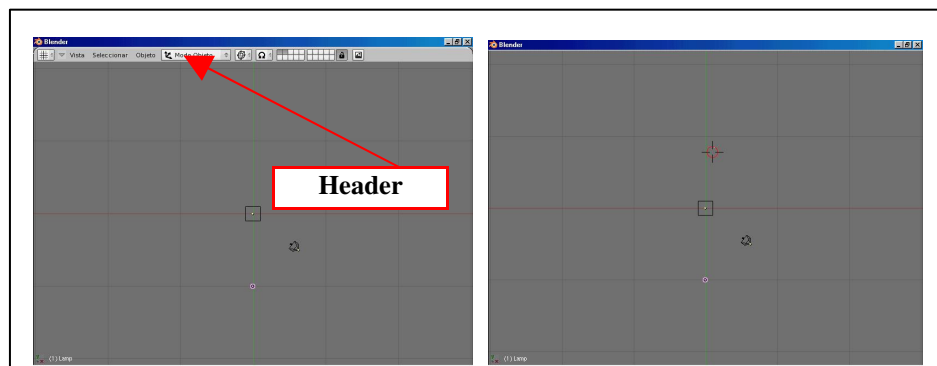


Figure 6. Window header.

The windows work as a cell that they can divide vertically or horizontally, to create this way a combination of windows of diverse types that adapt to our intention. Next it appears to a horizontal division and the vertical one of the window of the previous image.

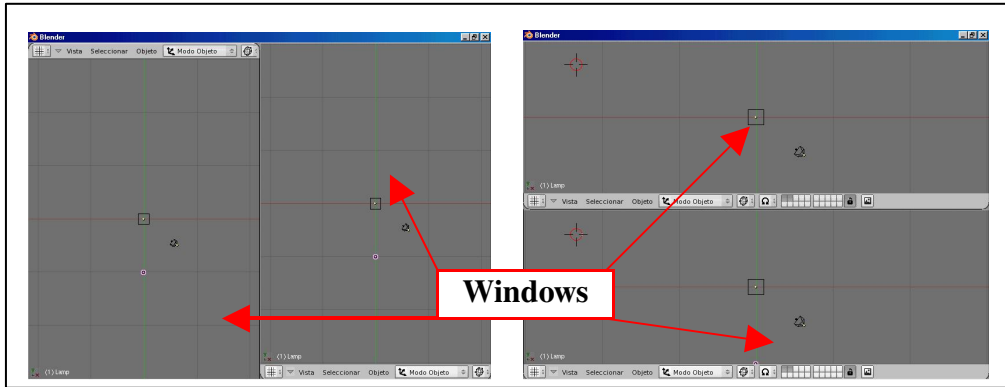


Figure 7. Blender windows.

To create a vertical divisions the mouse cursor must be placed in the border of the window until it changes to two opposite arrows up-down or left-right and clicking in the right button we will open the floating menu with the options to split, to join, to commute the visibility of the header and set the position of the heading.

These configurations of windows that we make are named SCREEN and it can be stored to be used later along with other screens configured by the user. Blender comes with 5 default screens with easy access making use of the keyboard (CTRL + right/left arrow). These are: Animation, Model, Materials, Sequence and Scripting that are to make the work easier for the tasks related to their title.

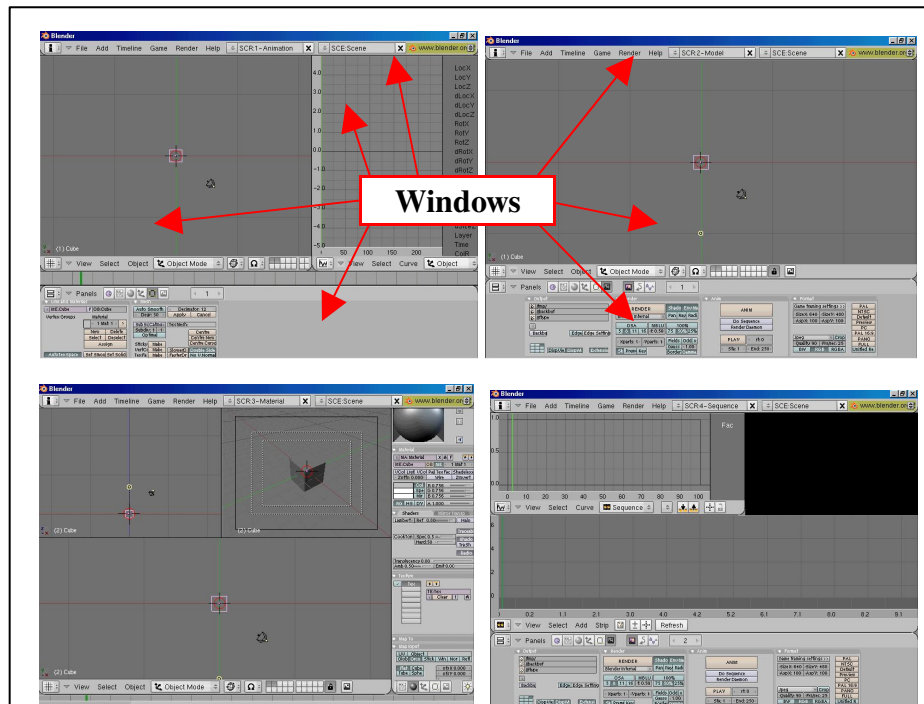


Figure 8. Blender windows.

It is also possible to commute the window between its defined size and full screen using the keyboard (CTRL + up/down arrow).

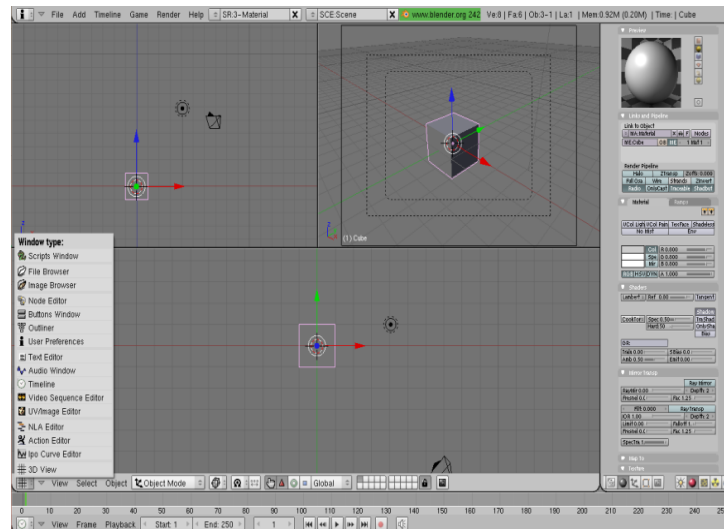


Figure 9. Window type selection menu.

Next we will name the types of windows that can be selected using the window type selection menu as it is shown in figure 9, and the ones used in this course will be explained ahead.

1. Scripts window
2. File Browser
3. Image Browser
4. Node Editor
5. Buttons window
6. Outliner
7. User preferences
8. Text Editor
9. Timeline Audio
10. Video Sequence Editor
11. UV Image Editor
12. NLA Editor
13. Action Editor
14. Ipo Curves Editor

15. 3D View

Sections and Panels

The panels are elements that group functions which interface could be alphanumeric or graphic. We have an example of it in figure 10. The panels are versatile, since they have an icon in the shape of an arrow located in the top left that makes possible to resize the panel, they can be moved as any objects in the 3D window and they can be grouped into one ordered by tabs.

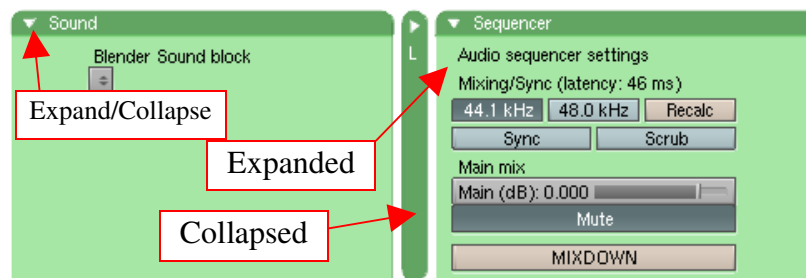


Figure 10. Panels.

The *context* are buttons that group panels and *sub-contexts* inside the same window of buttons. These are shown in figure 11 with their name.

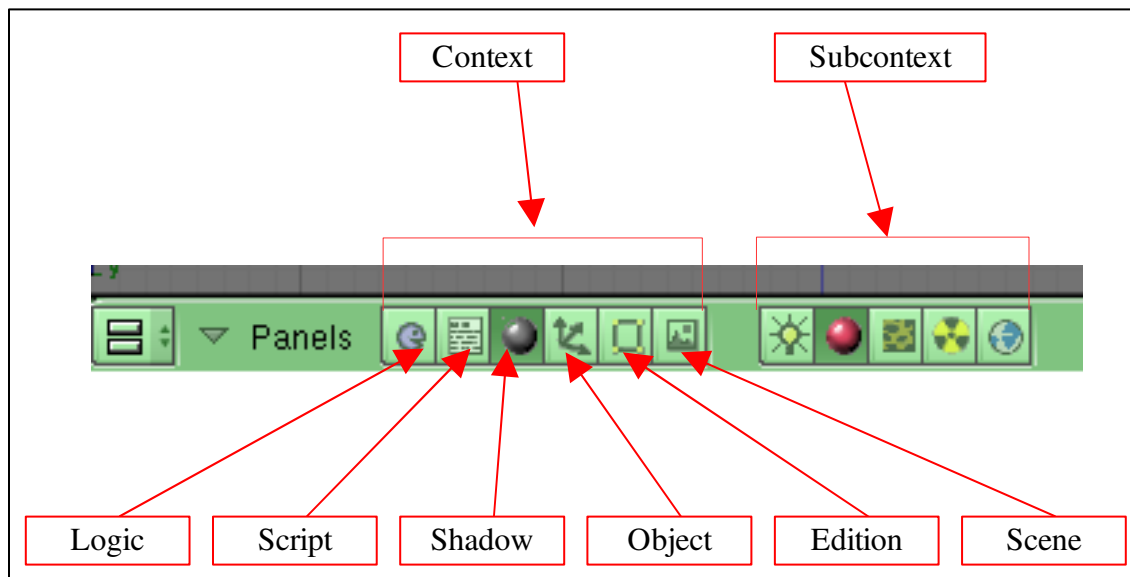


Figure 11. Blender Context and sub-contexts.

Toolbox

The toolbox is a floating menu that will appear when the space bar is pressed or any of the mouse buttons for more than 1 second in the 3d view window. It contains options like: View, Select and Object, that could be found in the header. And the others are to add new objects and to perform operations relative to the chosen objects, these are: Add, Edit and Select. The last one is for render options.



Figure 12. Toolbox's reference image.

Scenes

This is another revolutionary concept of many that Blender have. Starts from the idea of using different spaces for different shots, models and sequences that are a part of the same animation or from another completely apart, just as it is in the real life in a movie studio.

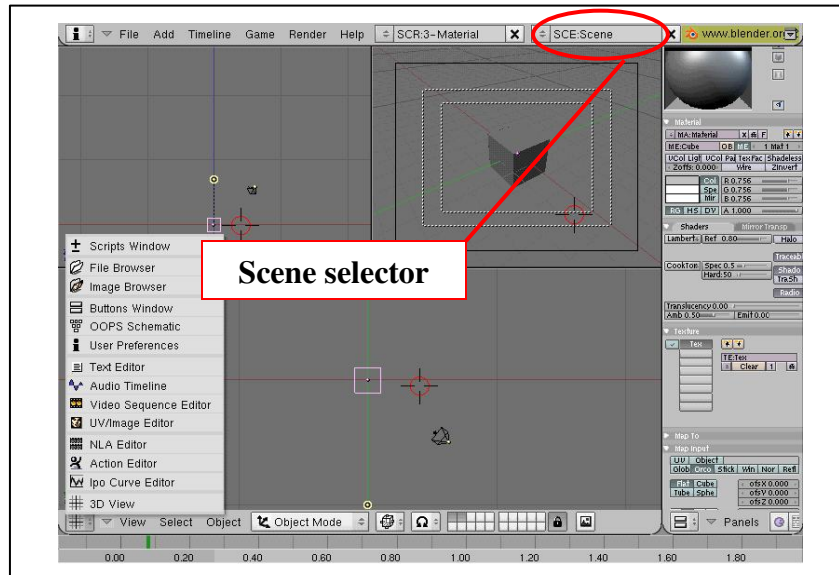


Figure 13. Scene Selector.

By means of the scenes selector is possible to create or eliminate scenes as well as to choose the one that is going to be used.

Navigating the 3D space

View Direction

The three-dimensional view is based on the cartesian system of coordinates that fulfill the law of the right hand, it means, the Z axis up as appears in figure 14a; each axis is differentiated by a color: green, red and blue for Y, X y Z axis respectively as appears in figure 14b.

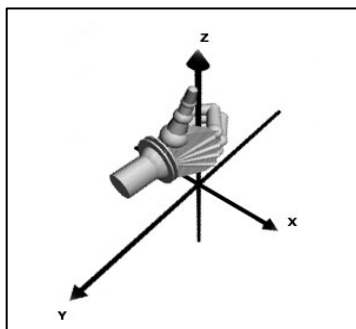


Figure 14a. Right hand's law.

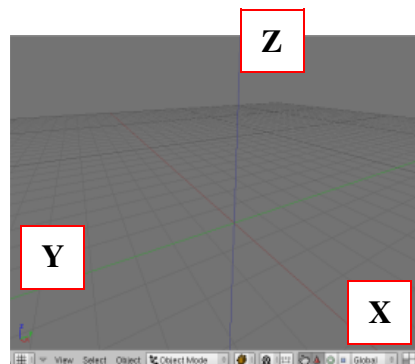


Figure 14b. Axis colors.

Views

The views are controlled by means of the numpad in predetermined positions:

- Up
- Down
- Left
- Right
- Front
- Back
- Perspective
- Camera

Pan, rotation and zoom in and zoom out:

These views can be (by default) rotated by means of the MMB + SHIFT and moved using MMB. Also it is possible to use the numpad to execute this function by pressing CTRL and 8, 4, 2, 6 to go up, left, down and right respectively.

Zoom in and zoom out could be carried out by clicking MMB + CTRL + dragging, or by using the mouse wheel.

Camera Projections:

The 3D window has two methods to project the objects: the orthographic and the perspective. The perspective is similar to the way we look normally the real world, where the nearest points turn out to be bigger than those who are farther away; but although this is the most natural, the default in Blender is the orthographic one, where an object maintains its proportions whether the observer is close or far from it. Although this one looks like a strange way of seeing the objects, it is the most precise.

Drawing Modes:

There are 5 drawing modes to work with:

- 1- Texturized. It tries to draw everything as complex as it is possible without coming close to the quality of details and time needed of the rendered image.

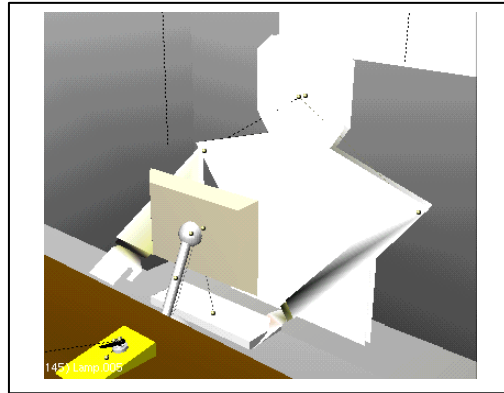


Figure 15. Texturized Mode.

2- Shaded. It draws the solid surfaces including the calculation of lighting.

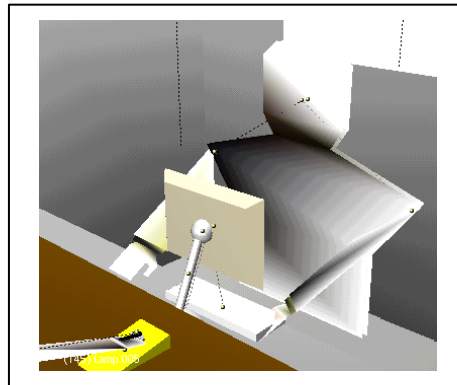


Figure 16. Shaded Mode.

3- Solid. The surfaces are drawn like solid without shades.

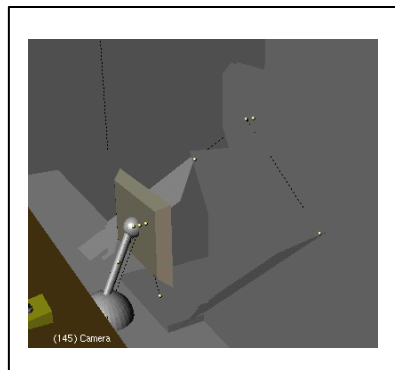


Figure 17. Solid Mode.

- 4- Wire. The objects are drawn by a few lines that allows its recognition.

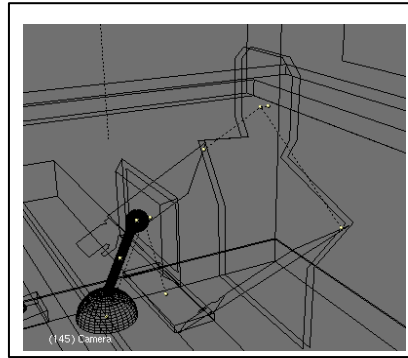


Figure 18. Wire Mode.

- 5- Bounding Box. The objects are not shown completely; instead of this, bounding boxes represent an object. This is the fastest of all drawing modes.

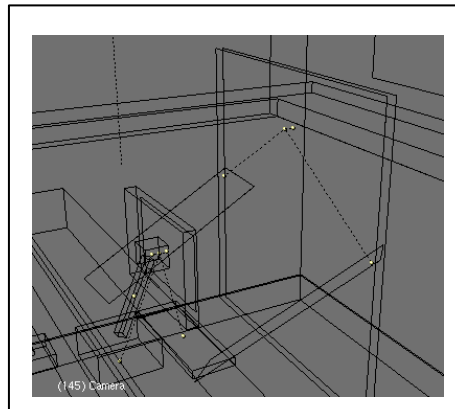


Figure 19. Bounding Box Mode.

To select a drawing mode click on the button that it is located in the header of a 3d view as indicated in figure 20.

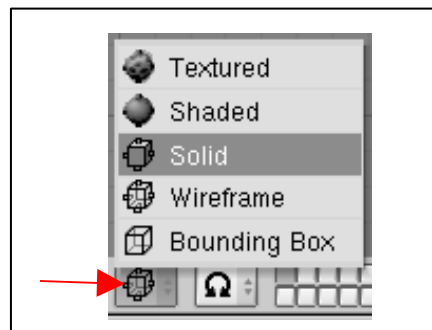


Figure 20. Selector of Drawing Mode.

The 3D cursor

This cursor is an indicator of a specific point in the space and it is represented as in figure 21. It changes to a new position by means of the LMB and is used among other things, to indicate where the new objects will appear when added to the scene or as the rotation point of objects.

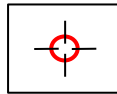


Figure 21. 3D Cursor.

Layers System

The layers are a set of buttons in the header of the 3D view window that are used to organize the visibility of the objects in the scene. This system is useful to develop complex scenes where it is necessary to conceal portions of the visual composition in order to have a lighter scene.

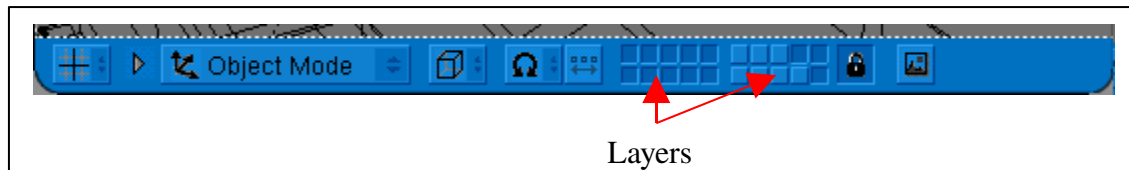


Figure 22. Layers Button in the header.

Click in the layer buttons to select the one that will become visible and Shift + click to select several at the same time.

To switch an object to another layer, press M in the keyboard, a floating panel will appear, select the layer and click OK. To cancel only move the mouse pointer out of the panel.

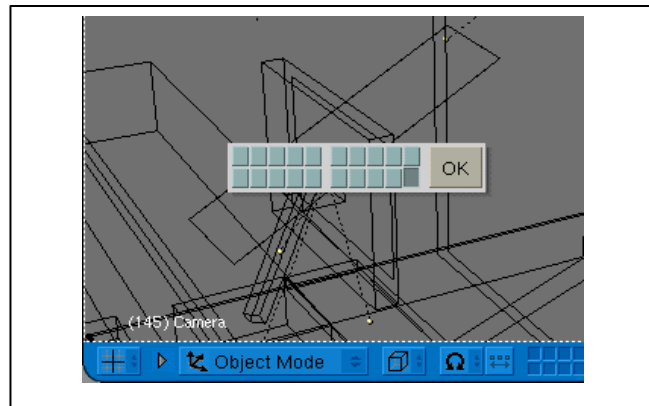


Figure 23. Panel to switch objects between layers.

Basic Functions

Loading and Saving Files

Blender identify the user working files with the extension .blend to store almost everything: objects, scenes, textures and even the configurations of window of the user's interface.

To gain access to the file browser press F1 and click with the MMB the file that you want to open. Valid file types will be marked with a color dot at the left side of the name of the file.

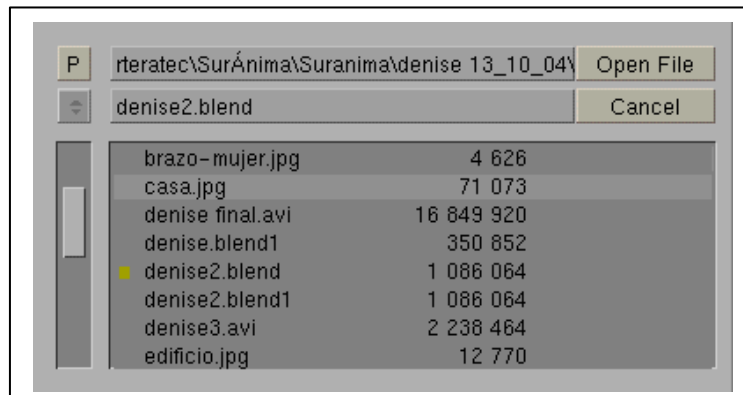


Figure 24. Load Files Window.

Saving files is a similar process, by pressing the F2 key, again the file browser will appear where the name and the path of the file to save could be specified. To quicksave without having to open any file browser press CTRL + W.

User Preferences

The administration buttons of the user preferences is located in the user's preferences window, which becomes visible dragging the low rim of the top window. These remain hidden due to its non frequent use. The use of each of the options contained in this section is self explanatory, so, only the main buttons will be explained.

View & Controls

In this section could be specified how Blender should react to the user entries.

Edit methods

Allows to specify the details to work with commands of edition like “duplicate”.

Language & Fonts

Select the font type and the language that it will be used.

Themes

Specify the graphical details in the user interface.

Autosave

Here are configured the characteristics of automatic backup, in case that something goes wrong, a safety copy is always kept and constantly saved.

System & OpenGL

If there is any problem with the graphics or the sound, this section will allow to tweak some characteristics that may solve them.

File paths

Here is where the paths for textures, fonts, sounds, plugins, python scripts and temporary files among others, are defined.

Modeling

Objects

All the objects that are wanted to be placed in a scene in Blender could be found in the toolbox under the option "add". Objects in figure 25 could be added under the mesh menu entry .

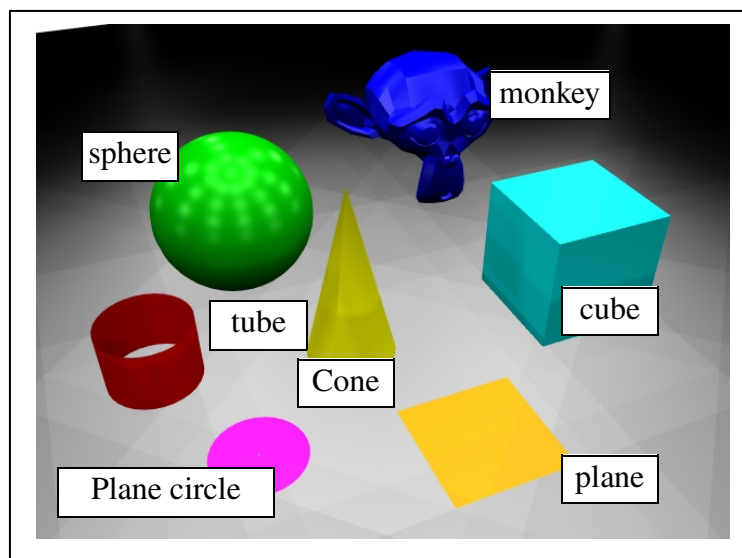


Figure 25. Primitives.

Every object has a pivot or central point or reference, which will be used as the local reference point of the vertexes that conform the shape of the object, among other uses. When a new object is placed in the scene it's pivot will be centered at the 3D cursor.

The manipulation of the characteristics of the objects begins on the selection of the work mode, depending if the manipulation is related to the colors of the texture, colors of the vertexes, manipulation of faces, manipulation of vertexes or modification of the object as such. As soon as the object is created, this one will automatically enter on *edit mode* which can be commuted to object mode using the key TAB.

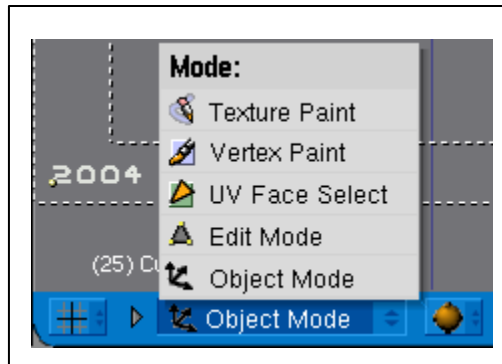


Figure 26. Operation Modes with objects.

We will begin with the basic actions in edit mode.

Select, scale, rotate and move

Using the RMB when the mouse cursor is over the object that we wish to select, this will become identified for having its edges highlighted in a light purple color (default color). In case there is superposition of objects, click repetitively until the desired one is selected. Repeating the previous action but this time pressing the CTRL key several objects can be selected at the same time, although only one can be active and this one will differ from the rest of the selection with a lighter color. The fact that an object is active means that is the one selected in the selection in order to visualize its characteristics in the buttons window. In case it is needed to select the objects located in an specific area of the screen, use the B key for “border selection”, the mouse pointer will change from an arrow to a cross where clicking and dragging an imaginary rectangle will define the selection area.

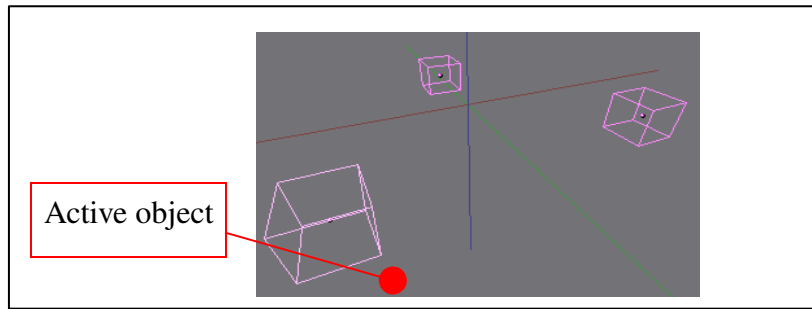


Figure 27. Selected objects and active object.

Once we know how to select an object, we will be able to change its dimensions pressing the S key and dragging the mouse closer or farther from the reference point, this one is selected in the header of the 3D view just to the left of the layer control and can be: the 3D cursor, the middle point of the selected objects, the center of the box that is delimiting the selection or the center of each object causing an individual scaling.

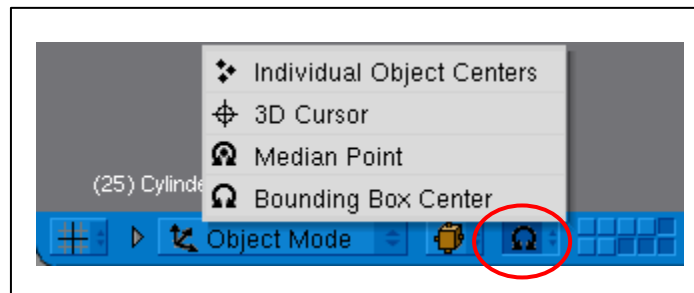


Figure 28. Selection of the reference point to scale and rotate.

If limitation of scaling is needed in an specific axis, press the X, Y or Z key in order to choose the axis of deformation and pressing it two times the local axis of the object is selected. It happens in a similar way with rotation; for this one the R key and dragging the mouse around the pivot it is used. And finally the displacement of the object is accomplished by using the G key and following the same rules as in the other cases.

Duplicate

In order to duplicate an object use the SHIFT + D combination, allowing this, to create an object of the same type as the active one, copy the characteristics and attributes, and select it in displacement mode. This function could be used to save design time at moment of placing series of objects.

Parenting

To link an object with another means to create a connection that relates primarily spatial properties. This is carried out selecting two or more objects and pressing CTRL + P. The connection will be made from all the selected objects (children) to the active one (father) and the connections will be indicated by dashed lines toward the father object as it shown in figure 29.

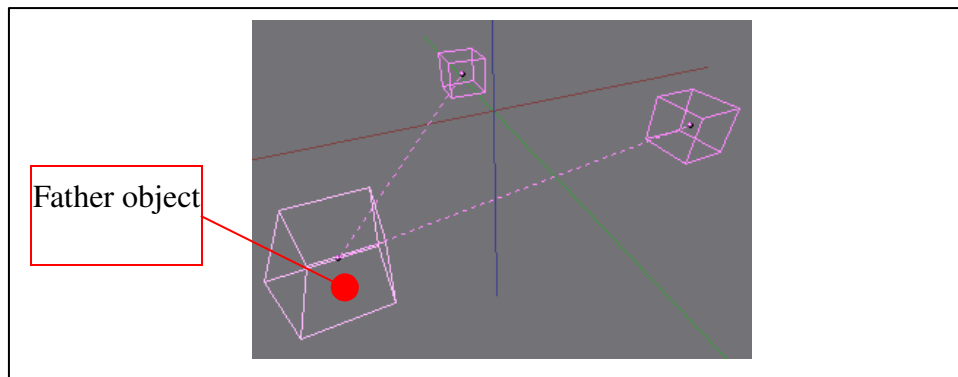


Figure 29. Objects links to a father object.

Meshes

When a 3d view is changed to edit mode, it changes the way that the selected object is presented to the user. Here the vertexes are shown for its direct manipulation in an explicit way. This combination of vertexes and edges conform what it is known as a the object mesh. As it is shown in figure 30, the selected vertexes are represented with yellow color and its selection and displacement are the same as for an object in edit mode, but the rotation and the scaling can only be carried out in groups of two or more vertexes. The border selection also applies to the vertexes in edit mode plus another characteristic, pressing twice the B key will enter *brush mode*, here the cursor will change to a dashed circle, which diameter can be modified using keypad NUM+ or NUM-, and the selected vertexes will be the ones under it while pressing **LMB**.

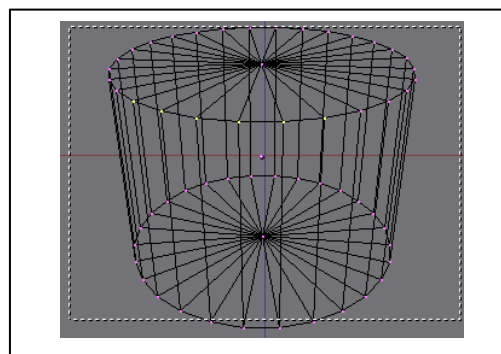


Figure 30. Vertexes selected.

For vertices manipulation it is necessary to know some basic functions. The first and simplest is the reflection. This is applied to the selection using M key where a floating menu will appear to specify the reflection axis.

Another important function to work with meshes is the extrusion, with which we can create an extension of a selected vertex or a group of these and the E key is used for that purpose. Figure 31 shows an example of an extrusion of a group of vertices.

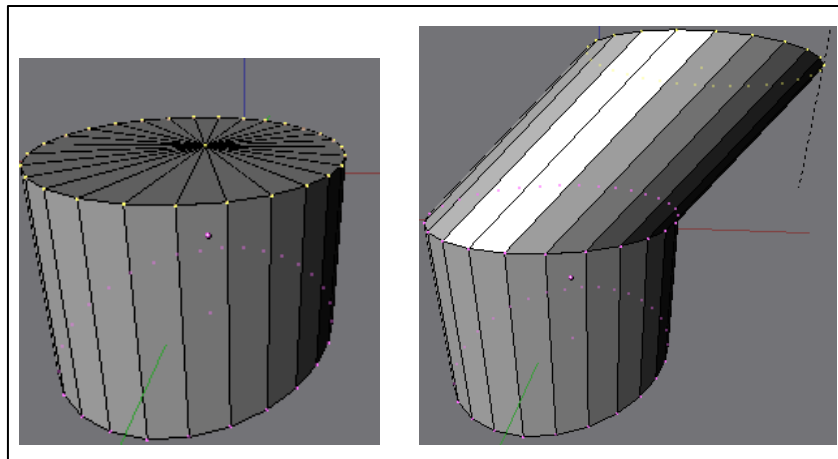


Figure 31. Cylinder (left) with extrusion (right).

The rest of the functions and properties for the manipulation of meshes can be found in the buttons window under the context *editing* () as it is shown in figure 32.

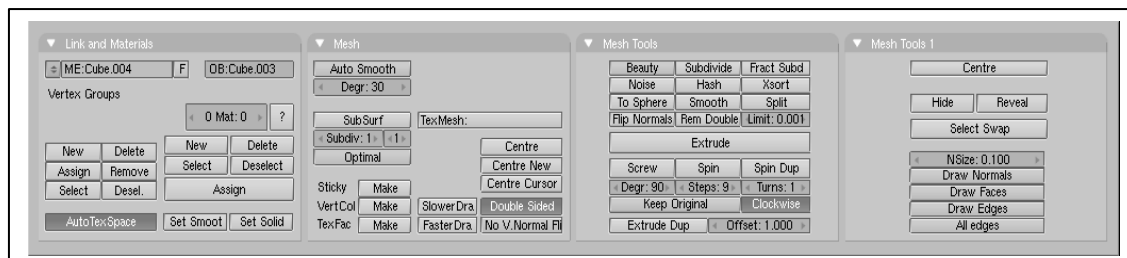


Figure 32. Buttons window, context *editing*.

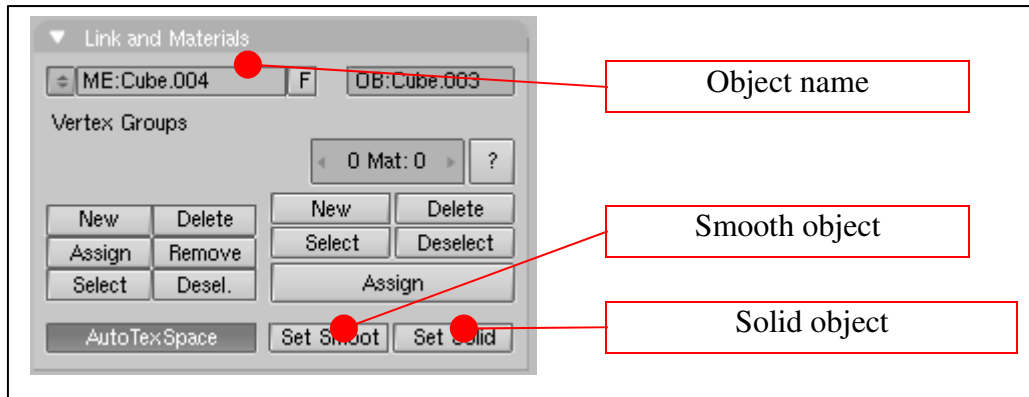


Figure 33. Links panel and materials.

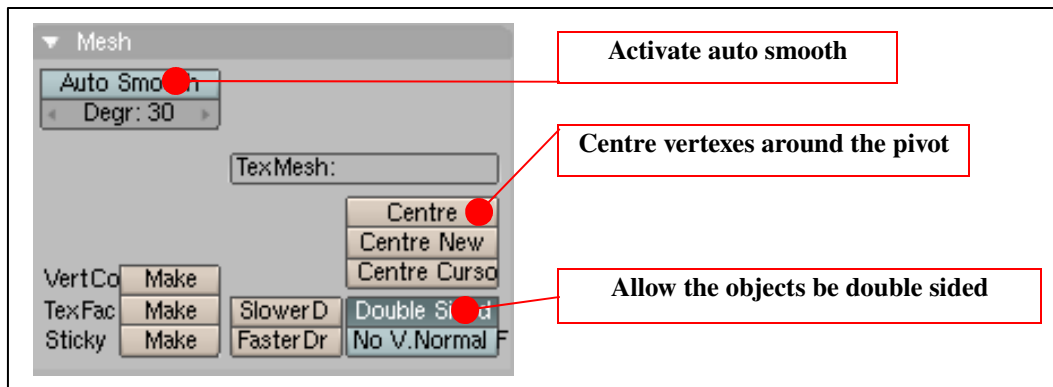


Figure 34. Mesh panel.

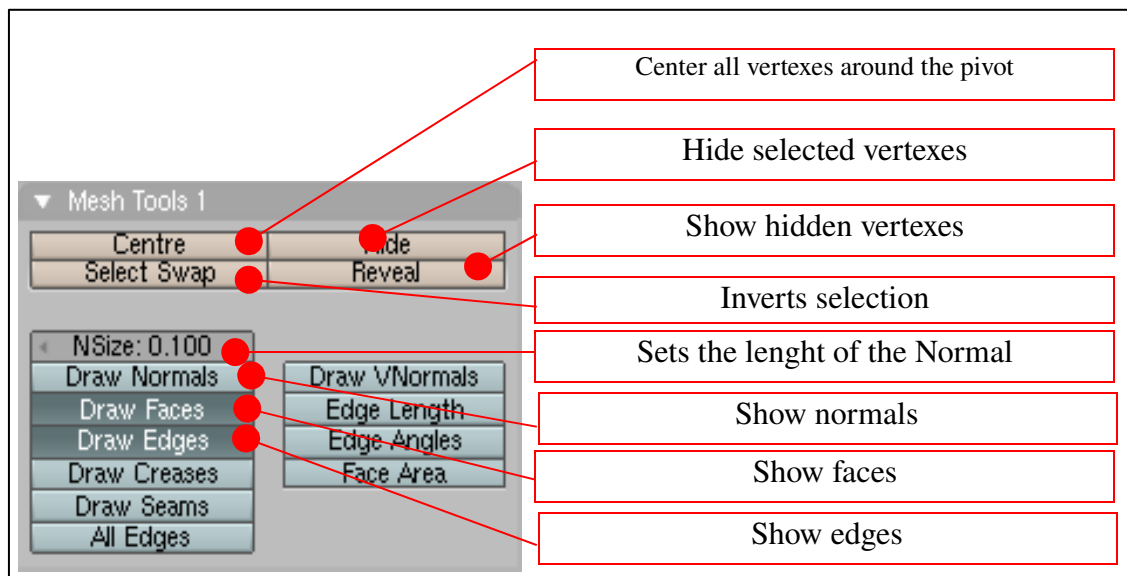


Figure 35. Tools panel of mesh.

Using the W key the special functions menu will appear:

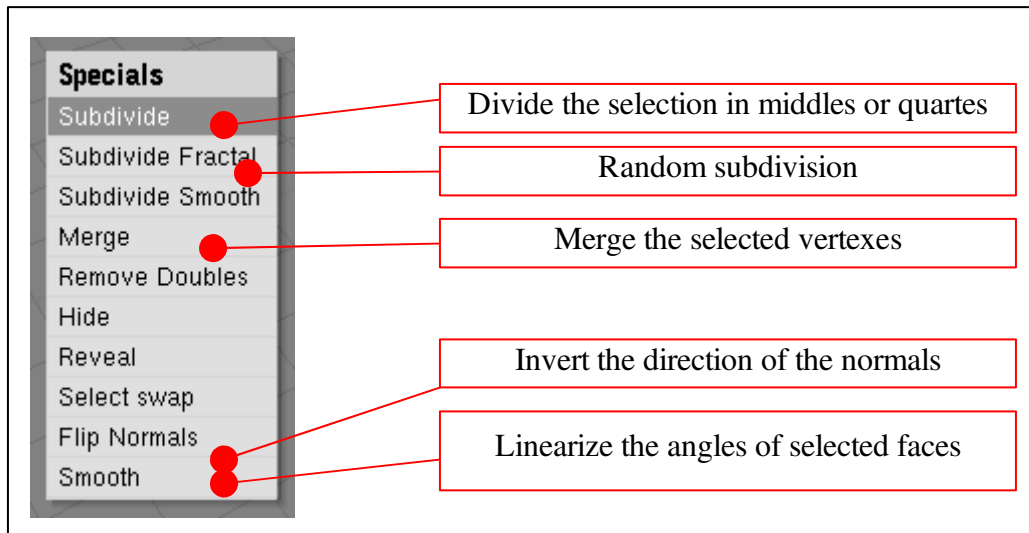


Figure 36. Floating menu for mesh editing.

Finally there is the undo function, which is implemented by means of the U key and it is used to go back N amount of actions done by the user, the value of N can be set in the user preferences window, specifically in the "Edit Methods" section, being 32 the default value. An alternative to use the undo function is the floating menu for that purpose, available pressing **ALT+U**, where the changes made throughout the time could be visualized in a list. In this same menu is also the function "Original", with which it is possible to be return to the beginning of the work in that session even though that point is outside the reach of the levels of undo.

Curves

To draw three-dimensional curves, Blender uses two different models, beziers and the Non Uniform Rational B-Splines or NURBS. Although both of them work with vertexes control and define a control polygon, the mathematics behind each are totally different where one is based on the interpolation of its control vertexes and the other in the attraction that exist between the nurbs objects in the set.

Bezier Curves

It is the most common type of curve, since it will be found whether as a 3D object, a path for other objects to follow, a deformation guide or even in the IPO animation control curves.

The control vertexes along the line defining the curve have two "handles" (control points linked to the control vertex) which are the ones to be used to shape the curve. These can be rotated, displaced and scaled as it would normally do with the vertexes in a mesh. They are defined in 4 types that are selected with the keyboard or automatically after changing the position of any of the handles. The types are the following ones:

- Free (black): Selected by hitting the **H** key (it commutes between free and aligned), it allows to control each handle independently.
- Aligned (purple): Selected with the **H** key (it commutes between free and aligned), locks the angle between the handles to 180° .
- Vector (green): Selected with the **V** key. Here each handle points out to the next control vertex.
- Auto (yellow): Selected with the **SHIFT-H** combination. The handles are modified automatically to give the highest degree of smoothness to the curve.

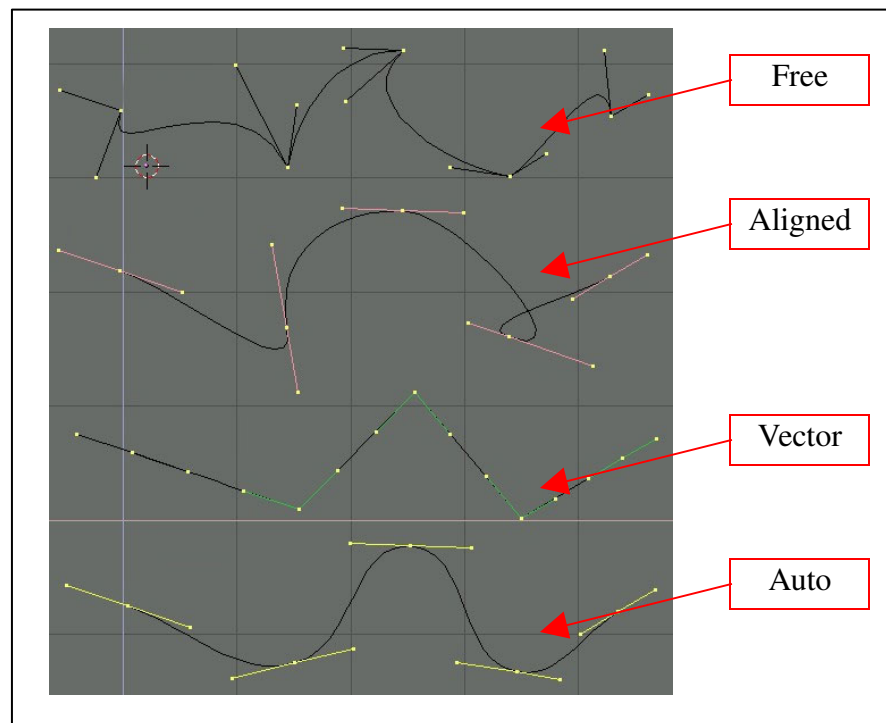



Figure 37. Types of vertexes alignment.

In the buttons window, context editing () , panel "Curves and Surfaces", there will be the variables to define the resolution that will have the curve along with other numerical control variables for the curve, like the level of extrusion or the resolution of the

beveling, among others. To set the resolution it is necessary to change the value of "DefResol" and then click in the button "Set" that is next to it.

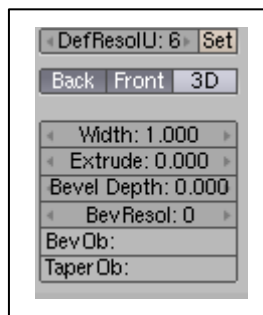



Figure 38. Curves parameters.

Text

The text object is a specific case of curves that resembles a standard font set and by means of its panel all the characteristics such as space between lines or angle of tilt can be modified. This panel is located in the buttons window, context editing () , where it is also possible to change the font type (TTF).

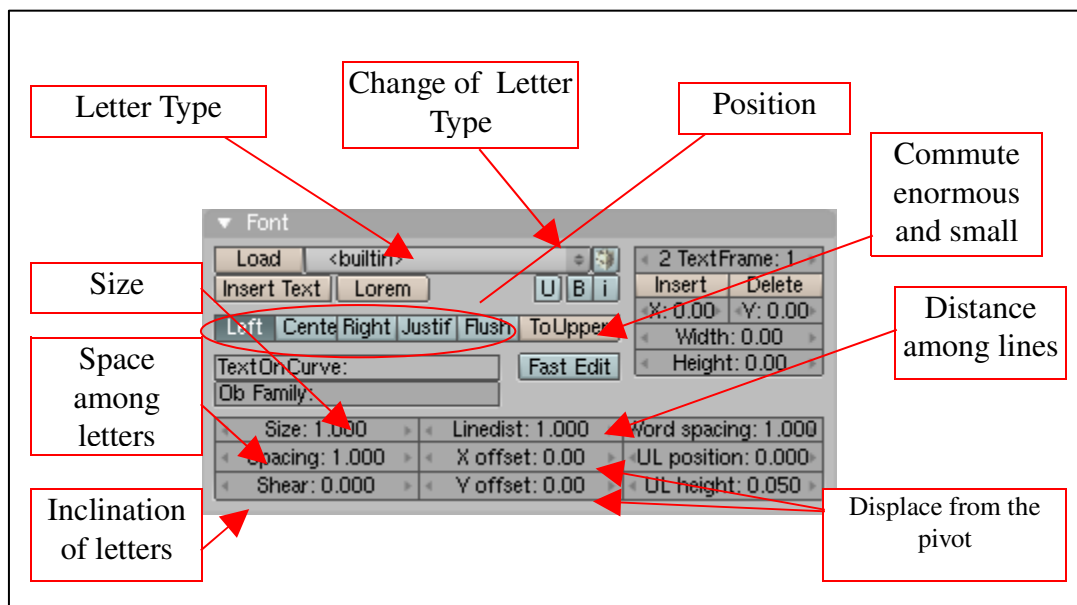


Figure 39. Configuration panel of typography.

Pressing the TAB key, it enters edit mode where the text can be modified as in any standard word processor.

Rendering

Rendering a scene is achieved by means of the "Render" button located in the *buttons window*, context *Section* (F10 key), sub-context *render buttons*; and the options are configured in the Output, Render and Format panels.

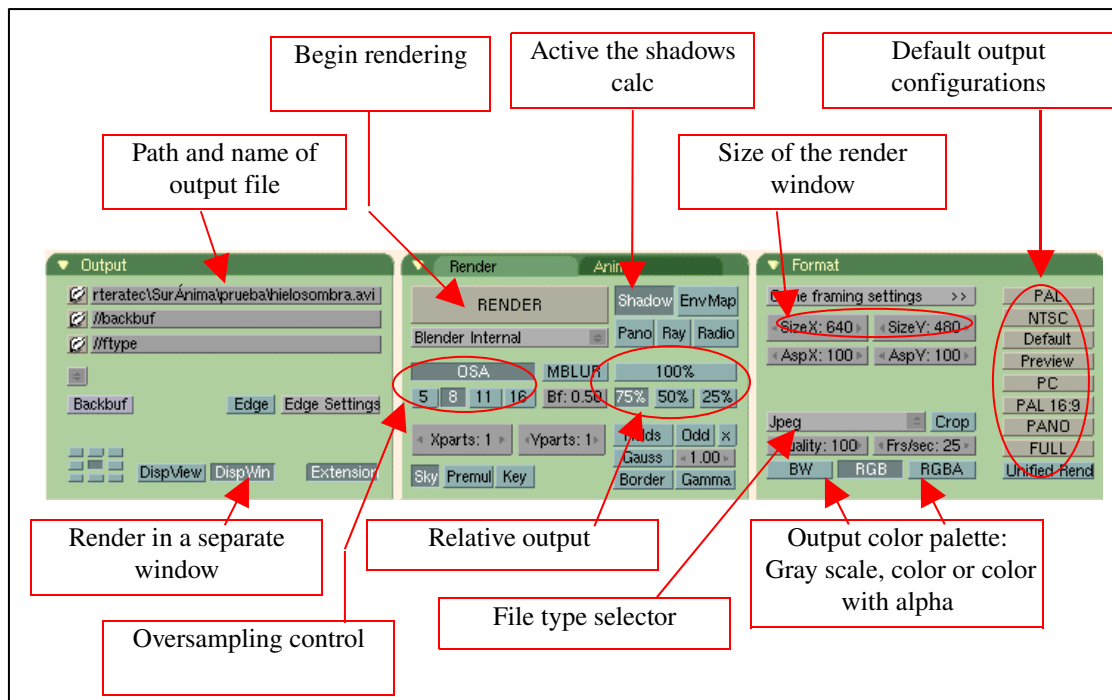


Figure 40. Render panels.

In the upper section of the panel *output*, there is a text box where the name and the path of the file to be save can be specified. In the panel *format* there are the buttons to change the size of the render window (in pixels) and a drop down menu to define the output file type; i.e : targa, jpeg, bmp, avi, etc. There are also the buttons, named BW, RGB and RGBA, that set the palette type to grayscale, color or color with alpha, respectively. Then at the right side of the panel, there is a series of buttons with different predefined configurations, as for example PC, which aspect ratio and output sizes are appropriated to to be used in a computer screen. In the same way there is the NTSC button, that it is designed to display the animation or the image in NTSC television systems .

Finally there is the Render panel, which it will be used to initiate the rendering process by means of the Render button. Also in this panel it is possible to activate the scene shades calculation and to define the relative output size. This defines in percentages, the relative size that will have the rendering window related to the dimensions established in the format panel. This is especially useful when previews are needed frequently, because the less the output percentage, the faster the rendering process it will be. At last and to

finish this section, a set of five buttons is located at the central left side, being the biggest the one called OSA, this button is the one that allows to activate or to deactivate the oversampling of the scene, eliminating sharp edges of the image and it could be applied in steps of 5, 8, 11 and 16 in the respective buttons. Figure 41 shows to the left a rendering image without OSA and to the right with OSA level 16. Look at the rims of the sphere and the plane.

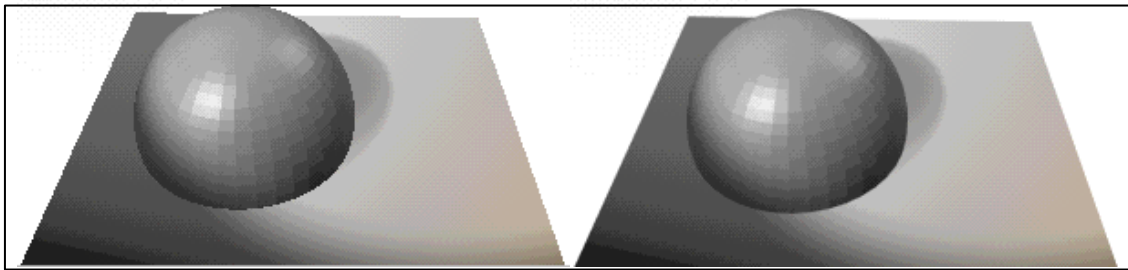


Figure 41. Left: Image without OSA. Right: Image with OSA.

Materials and Textures

Technically, materials and textures are data structures associated to the objects. These structures are containers for characteristics like: the color, the 2D shape, the transparency, etc, that every object must have in its “skin”. To better understand the concept of material and texture, let's think about the analogy with a real life clay statuette, where the clay is shaped to achieve the wished form and then it is painted, to give it the colors that represent for example, the cape of a saint or the eyes of a doll.

Materials

Although more than one material could be assigned to an object in a 3D scene by means of advanced techniques, we will assume that it is just one, concerning simplicity.

By means of the material, Blender is able to define how the “skin” of the object will react to light, being able to define characteristics like color, transparency, reflection, specularity and quantity of light that it emits. The controls for these characteristics are located in the buttons window, shading context, materials buttons sub-context, and it is divided in 4 panels: Preview, material, shaders and mirror transparency.

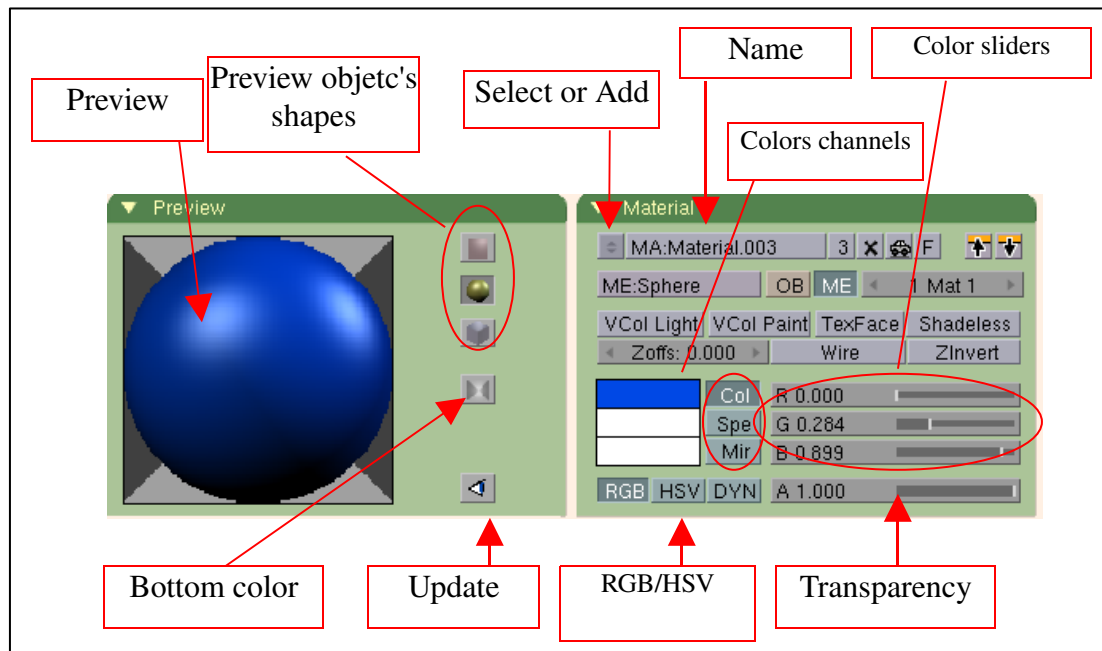


Figure 42. Panels: preview and material.

A defined area in the preview panel displays the actual visual properties of the material and depending of the selected shape: a plane view, a sphere or a cube can be seen.

In the "material" panel, the most basic characteristic is the color channels. These are: the base color, the specularity color and the reflection color, and by selecting each one of these, it allows to modify its values in RGB or HSV. Also we will find in this panel a slider to change the value of the alpha channel. This is the one that it will define the transparency of the object, 0 totally transparent and 1 totally opaque.

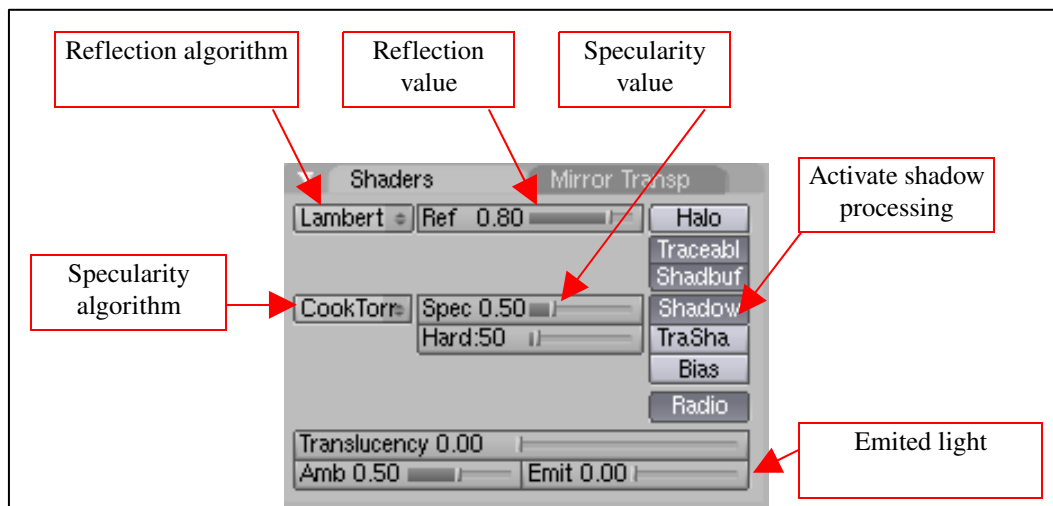


Figure 43. Shadow panel.

We have mentioned in several opportunities a material property called “specularity”, now we will define it. The specularity is the reflection property of the material taking into account the observer's position, because it calculates it by means of the Snell laws, where is dictated that the light's refraction angle will be equal to the incident angle. In the practice it will give us a sheen property, **whether be diffuse or sharp** on the surface giving us the sensation of rough or polished surfaces. Along with the specularity or specular reflection, there is also the diffuse reflection that is similar to the first one, but it is independent from the observer's point of view.

In the panel “shaders” we will find a pulldown menu to select the algorithm to calculate the diffuse reflection, these are: Lambert, Oren-Nayar and Toon. The first one takes into account the color of the material and the quantity of energy to be **spreaded**, the second is the same plus a third parameter to determine the microscopic roughness of the material and the third algorithm is to create a cartoon effect. Just like before, the algorithms for the specular reflection are: CookTorr, Phong, Blinn and Toon.

Continuing in the same panel, we will find the “shadow” button that is to commute the reception or no reception of shadows by that material and at last, on the bottom right, the slider to set the value of the light emitted by the material, if a different value from zero is used, will make that object visible even though there were no lights in the scene.

Textures

When we see an object in the real life and look closer to its surface we will see that it has irregularities given by the color or the shape, like the skin of an orange, which is not smooth but rather porous. In our 3D scene it would not be sufficient to use a material, since with this only the color and the behavior against light could be simulated, it is necessary then to have another data structure that contains the information that we need to give the touch of realism that we are looking for. These structures are called textures and can be applied in layers.

In the buttons window, context Shading (F5 key), sub-context Material buttons; it is located the panel identified as "Texture" and as it is indicated in the figure 44 we can create a new texture or assign an already created one to the material that it is being used.

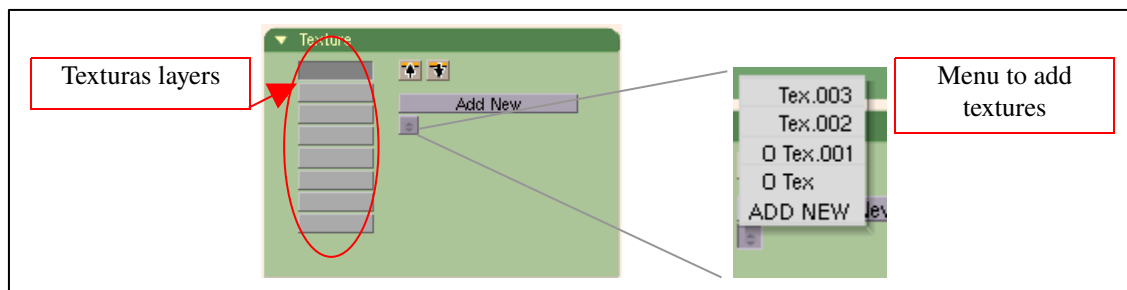


Figure 44. Texturas panel.

Later we select the Texture Buttons sub-context (F6 key) to define the characteristics of the texture. The actual texture will be shown in the preview panel and next to it there are the buttons to indicate whether the texture is for a Material, a World or a Lamp.

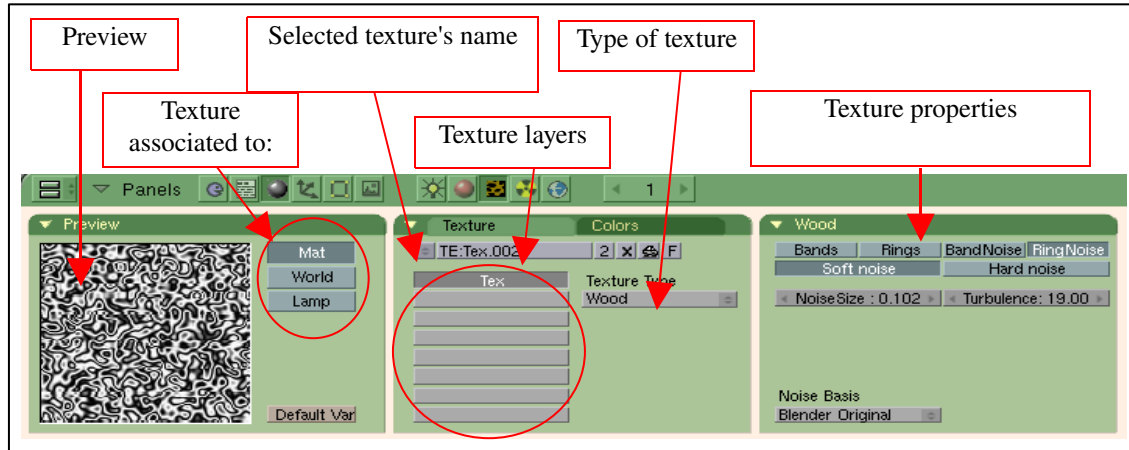


Figure 45. Textures configuration panel.

The panel called "texture" contains a series of piled up buttons that represent the texture layers, being the one at base the most external layer in the object in which it is applied. At the right part of the same panel it is a pulldown menu to select the texture type where is possible to choose (i.e): wood, stucci, clouds, noise, marble, distorted noise, voronoi patterns, etc.

Now, going back to the material buttons sub-context, we will find two more panels called "Map to" and "Map input". As it is shown in figure 45, the Map Input panel contains the buttons used to select the general shape of the object, wheter is flat, square, tubular or spherical, and it is especially useful when working with image textures, since with this selection certain modifications are applied to the image in accordance to its shape. Also controls are provided for the size and the displacement of the texture in each of the axes. On the other hand, the "Map to" panel, it contains among many other characteristics and values, the sliders to select the texture's output color and the intensity of it.

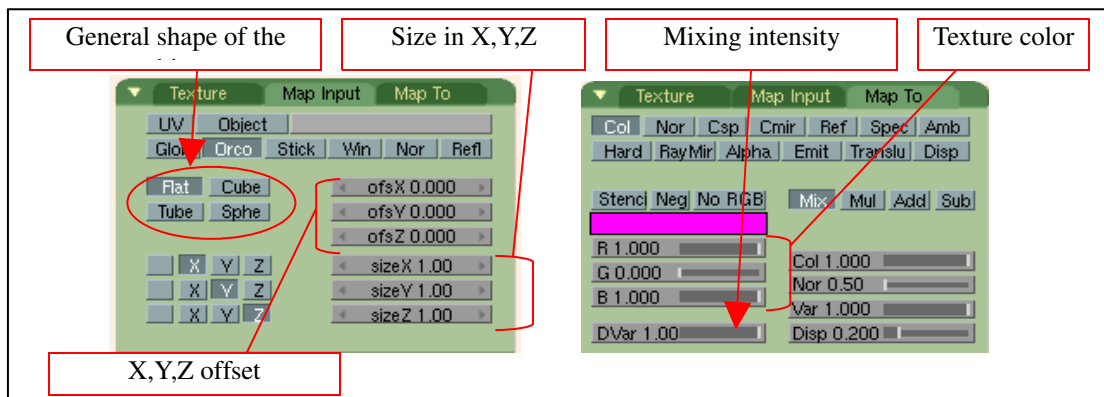


Figure 46. Textures configuration panel.

Once the material and the texture to apply to the active object is selected, we must render the scene to be able to see the final results of our work as it was mentioned in the section called "Rendering".

Lighting

Lighting is one of the major topics in the creation of 3D worlds and it would take a separate book to fully cover it, that's why we will only explain the types of lights and its general characteristics.

We must initiate then, by adding a light to the scene by means of the toolbox in the section "Lamp" and once it has been added, we will be able to see its properties in the buttons window, context shading, sub-context "Lamp buttons", similar to the one shown in figure 47.

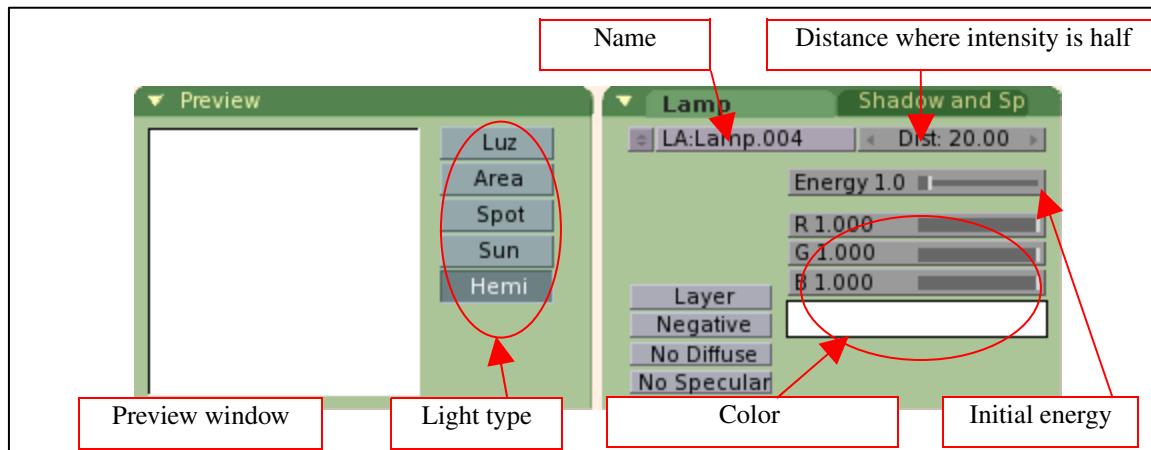


Figure 47. Preview panel and Lights main panel (Lamp).

In this section the preview panel contains an area to visualize the colors and the texture (in case of having it) of the chosen light, there are also buttons to select the type of light: Lamp, Area, Spot, Sun and Hemi. Next there is the panel "Lamp" where there are the basic properties for all types of lights, and in accordance with the chosen type, the specific options will appear. There are also the controls to change the value of the distance at which the energy of the light source is half of its initial intensity, the value of the initial energy and the color of the light, among others.

Just like the one before, the panel "Shadow and Spot" changes completely depending on the chosen type of light.

Focus

This light has the characteristic that it can be pointed to a specific direction and that is the only one that can cast shadows. Figure 48 shows the relation between the variables of the panel and its visual representation in the 3D window:

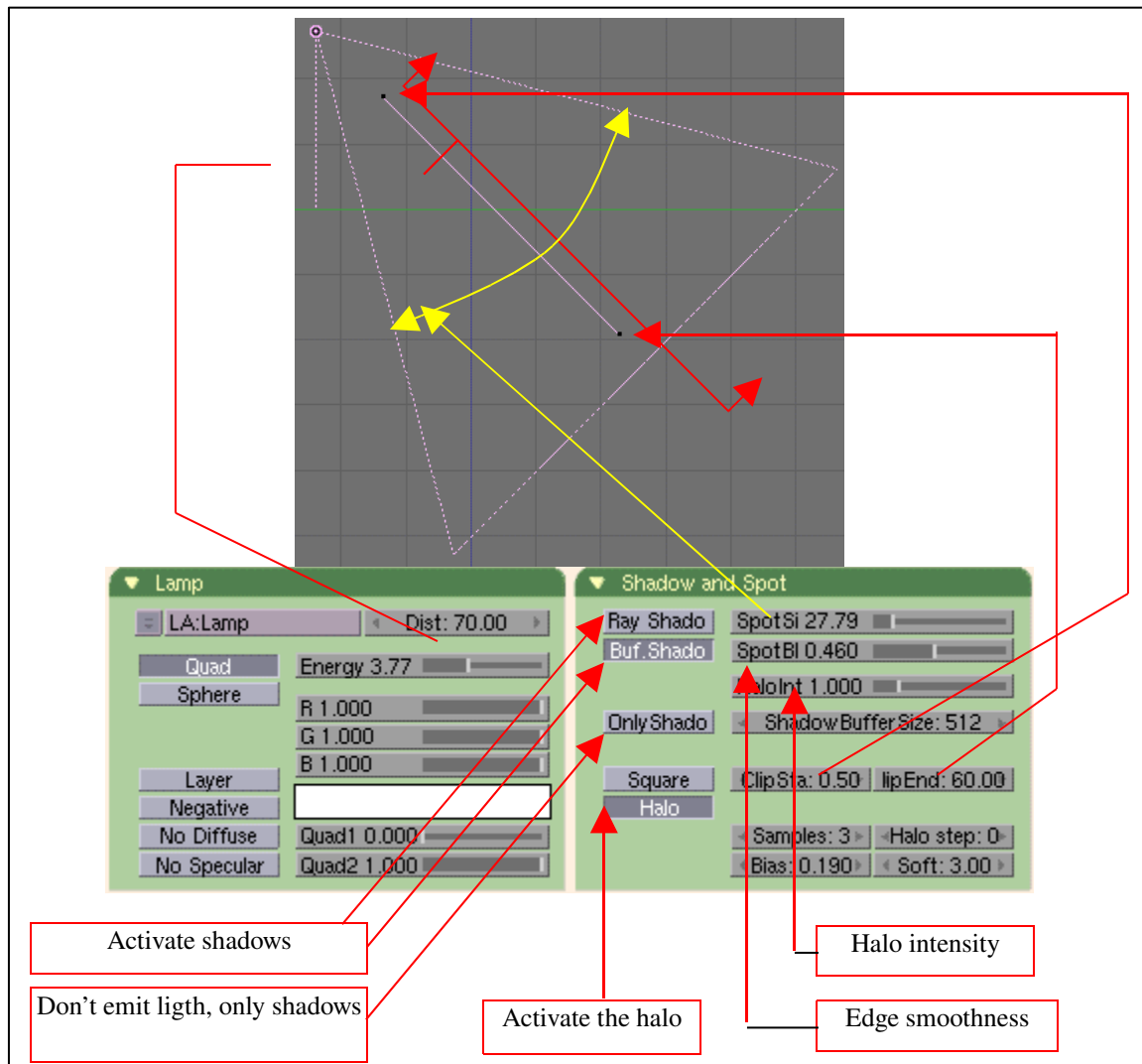


Figure 48. Focus configuration.

This light has a property called Halo, which allows to create the effect that create lighth in an foggy ambience and its intensity can be set in the “HaloInt” button. It also has the ClipStart and ClipEnd variables, which delimits the illuminated area of the cone of light.

Lamp

This is the default type of light when inserting a light in scene and is omnidirectional, this means that it radiates the same amount of energy in all directions from the emitting point.

Sun

It is a type of light that simulates the solar light, generating parallel beams in a specific direction at a constant intensity, that's why it is the simplest type of light. It is used frequently to simulate outdoor shots.

Animation

We have finally arrived at the most exciting topic in the 3Dcg world, the animation, where giving "life" to a simple cube simulating a speaking dice or a simple plane that simulates a dancing sheet of paper, is one of the most exciting experiences in this field of work. We have talked about objects, materials, textures and lighting, which is equivalent to the words of a film director: lights, camera there is just needed the action.

As it has already been mentioned, a video sequence is a sequence of stills that are shown one after another at a certain speed, thus generating the illusion of movement and definitions like frames per second (**fps**). Therefore, to make an animation it is necessary first of all, to set the fps and the length (in frames) that the animation will have. This task can be carried out in the sub-context "render buttons", panels "Anim" and "Format". In the first one there can be set the amount of frames that the animation will have by defining the initial and final frame in *Sta* and *End* respectively. Then by means of the "ANIM" button the rendering process will begin.

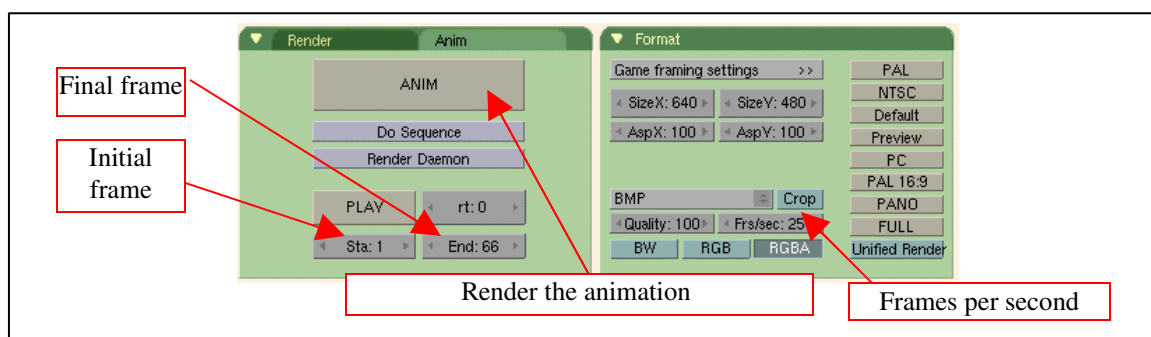


Figure 49. Configuration panel of animation render.

Located at the right side of the buttons window's header, there is a button with a number that indicates the actual frame, and in order to change its value, it can be done whether by clicking with the **RMB** and dragging, by using **SHIFT+RMB** to manually

specify it, or by hitting the arrow keys, where up and down arrows shift frames by 10 and left and right arrows shift frames by one.

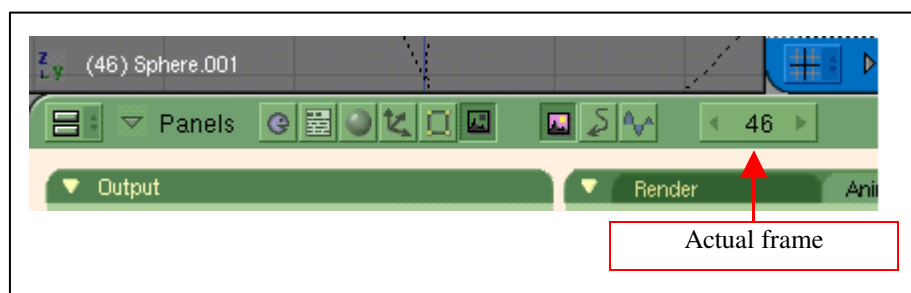


Figure 50. Selection button of frame.

Following, we define the type of output file in the format panel. We must define a type of output file of video as it is AVI (audio video interleave) and with this the platform of output is already defined for our animation.

Keys

The keys are the most important concept of an animation. These are the structures of information that keep the information of an object, for example its position, rotation, size, etc., even the color of its material; all of it associated with a specific frame.

Using the **I** key, there it will appear a menu for the insertion of keys in Blender similar to the one presented in the figure 50. In there we can find the type of key to insert, if it is a key to define position, rotation, size, etc. The keys exists due to how Blender animates the objects, the transitions are planned first, for example a box is to move from point A in the frame 1, to point B in the frame 29, we insert a key of position in the frame 1, change to the frame 29, displace the box up to the point B and insert again a key of position. All the intermediate positions that the box must take between the frame 1 and 29 are calculated automatically taking into account if the transition type is linear or Bezier.



Figure 51. Keys insertion menu.

NLA Editor

This window is the space where there are visualized in a graphical form, the keys that every object at a given time. It is similar to the one in figure 52 and the control on every key follows the same conventions of the manipulation of objects in the 3D window, that is, **RMB** to select, **G** key to grab and drag, **S** key to scale and so on.

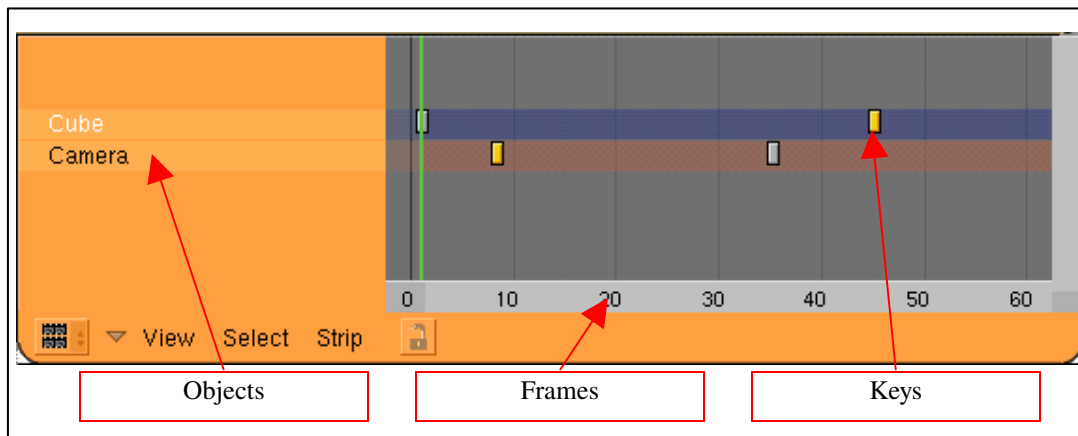


Figure 52. NLA editor.

IPO Curves Editor

The IPO curves are the accurate control tools of the changes between keys. Graphically here it can be specified the way that all the intermediate frames will be generated (the ones between the frames where there are keys defined), and the value of a property at a certain time. Its aspect is shown in the figure 53.

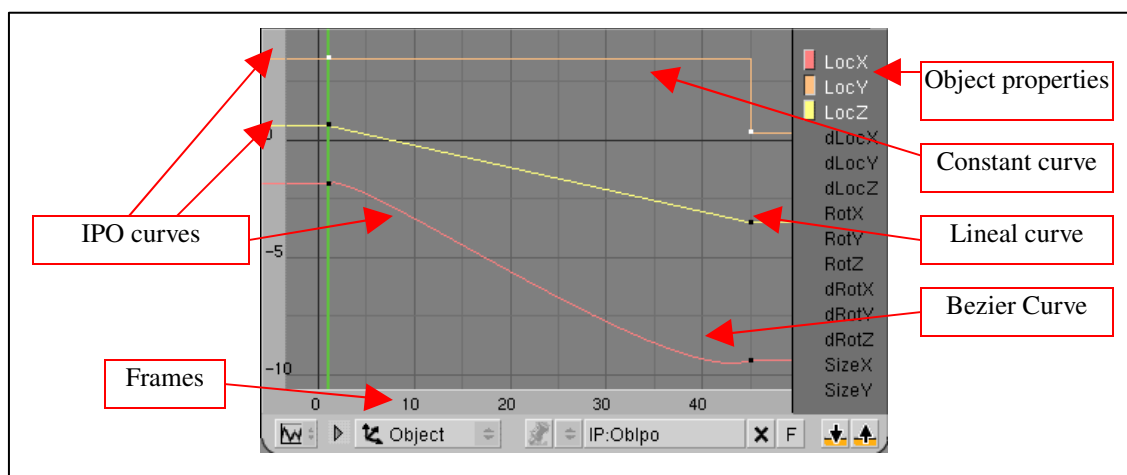


Figure 53. IPO curves editor.

The curves can be of three types according to its interpolation:

- Constant: There are sudden transitions among the vertexes, without generating intermediate changes.
- Lineal: The transitions between the vertexes are in a linear way.
- Bezier: The transitions among vertexes are smooth and controlled with the handles, similar to the curves of the objects.

There is another way to add a new vertex that is not by means of the menu. This one is by selecting the property of the object in the IPO curve editor and then **CTRL+LMB** in the position that the new vertex is to be added and therefore the new key, unless a vertex of another property is already there in the same frame.

How the curves behave after or before the final vertexes is known as the extend mode of the curve and some are: constant, where the value will remain the same all the time; and cyclical, where the pattern of the curve will repeat in an indefinite way between the first one and the last vertex.

The interaction with the curves is similar to that with the objects in the 3D window. If we do an analogy between both we can have the curves as the objects where the grab and scale methods are the same, and it is also possible to change to vertex edit mode.

